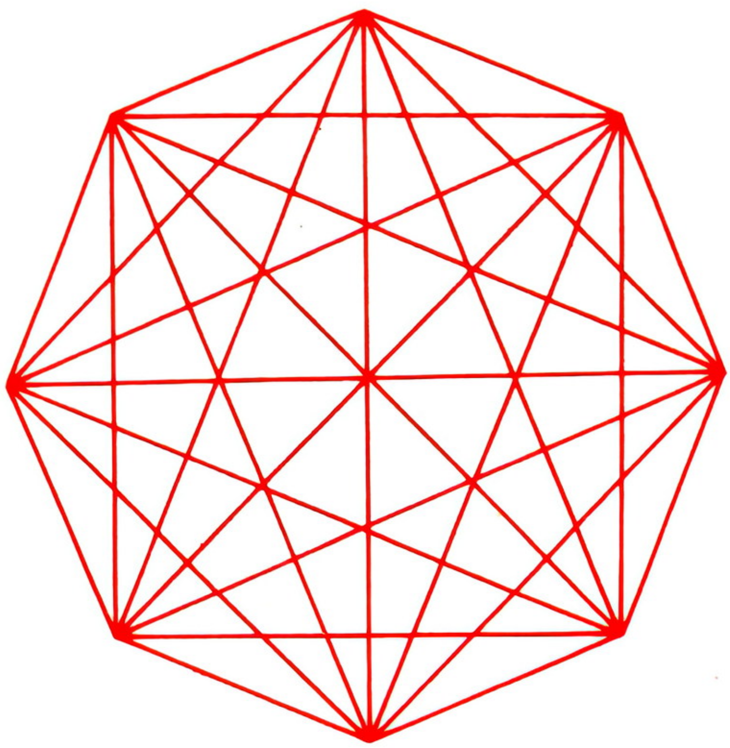
Axel Plenge

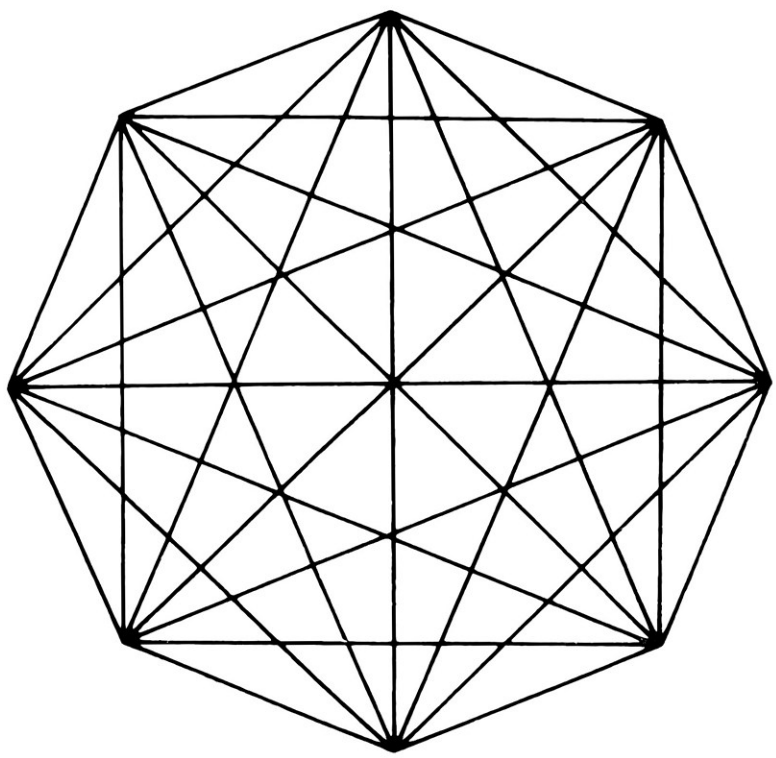
GRAFIKA
A COMMODORE 64-ESEN



DATA BECKER – NOVOTRADE

Axel Plenge

GRAFIKA
A COMMODORE 64-ESEN



DATA BECKER – NOVOTRADE

A könyv eredeti címe: Das Grafikbuch zum Commodore 64 (1985)

Fordította: APEX Szervező, Szolgáltató GMK

Lektorálta: DR. LENGYEL JÓZSEF

A kiadásért felel RÉNYI GÁBOR, a NOVOTRADE RT. vezérigazgatója
Budapest, 1988

Műszaki szerkesztő: Dévényi Erika

A szedés Commodore PC 20-as számítógépen és Hawlett Packard Laser Jet sornyomtatón készült a kiadóban

Készült a Somogy Megyei Nyomdaipari Vállalat kaposvári üzemében
(18 A/5 ív)

Felelős vezető: MIKE FERENC igazgató

ISBN 963 02 5637 1

FONTOS TUDNIVALÓK

A könyvben ismertetett kapcsolások, eljárások és programok nem tekinthetők szabadalmi oltalom alá eső ipari termékeknek. Ezek elsősorban amatőr és oktatási célokat szolgálnak. A szerzők rendkívül nagy gondot fordítottak a kapcsolások, műszaki adatok és programok helyességére, a részletek kidolgozása során többszöri ellenőrzést végeztek. Mindez azonban nem zárja ki az esetleges hibalehetőségeket.

Az előforduló hibákért és az ebből adódó következményekért a DATA BECKER cég sem szavatosságot, sem jogi felelősséget nem vállal. Az esetlegesen előforduló hibák közlését a szerzők hálásan fogadják.

TARTALOMJEGYZÉK

ELŐSZÓ 9

1. Fejezet – Bevezetés 11

2. Fejezet – Bitek és byte-ok 13

2.1 A tízes vagy decimális számrendszer 13
2.2 A kettes vagy bináris számrendszer 14
2.3 A tizenhatos vagy hexadecimális számrendszer 16
2.4 Logikai műveletek 17

3. Fejezet – Hardver alapismeretek 20

3.1 A VIC regiszterei 20
3.2 A VIC üzemmódjai 27
3.3 A CBM 64 tárolókezelése 31
3.4 A pontgrafika 45
3.5 A sprite-ok 53
3.6 Szöveg/jelkészlet 63
3.7 IRQ-lehetőségek 70

4. Fejezet – A grafika programozásának alapjai 79

4.1 Szöveg és grafika a kisfelbontású képernyőn 79
4.2 A pontgrafika programozása 89
4.3 A sprite-ok programozása 107
4.4 A jelkészlet programozása 137
4.5 A grafika tárolása és betöltése 157
4.6 Megszakítások 162
4.7 Kis grafikai programcsomag 171

5. Fejezet – Alkalmazási lehetőségek 191

5.1 A grafika alkalmazási területei 191
5.2 Futó feliratok 226
5.3 A játékok titka 229

6. Fejezet – Függelék 240

6.1 Az optimális program 240
6.2 A grafikus tár felépítése 242
6.3 Színkódtáblázat 244
6.4 Képernyőkódok 245
6.5 Konverziótáblázat (dec/hex/bin) 248
6.6 Sprite-tervező adatlap 249
6.7 Jeltervező adatlap 250
6.8 A VIC regiszterei 251

Előszó

A C 64-es számítógép egyik fő erőssége a grafika, amelyet azonban a Commodore cég alaposan elrejtett. A kezdők szájtátva csodálják a sok akciójátékban és kész programban előforduló grafikákat, amik a megfelelő rutinokkal – általában gépi nyelven – a készülék grafikai képességeit teljes mértékben kihasználják.

Ezzel a könyvvel most lehetőséget kívánunk nyújtani minden 64-es felhasználónak ahhoz, hogy saját programjaiban is kamatoztathassa számítógépe ez irányú tudását.

Szerzőként sikerült megnyernünk Axel Plenge urat, akinél jobban senki sem ismeri a Commodore 64-est, különös tekintettel annak grafikai oldalára. Mindezt a közkedvelt SUPERGRAPHIK-kal már be is bizonyította. Ezt a munkát is nagy lelkesedéssel végezte és közben annyi érdekes dolgot fedezett fel, hogy a könyv végül 50 oldallal bővebbre sikerült a tervezettnél.

Mindez csak az ügy előnyére válik. Sok sikert kívánunk tehát Axel Plenge programjainak és feladatainak kipróbálásához:

Dr. Achim Becker

1. FEJEZET

Bevezetés

"64 k RAM, 8 független, a képernyőn szabadon mozgatható sprite, nagyfelbontású grafika 320×200 ponttal, többszínű üzemmód, 16 szín, 40 oszlop, 25 sor, változtatható jelkészlet, szenzációs megszakítási lehetőségek, ... stb. Egy számítógép, amelyet mindenkinek ismernie, sőt alkalmaznia kell! ..."

Így, vagy ehhez hasonlóan dicsérik a mi jó öreg Commodore 64-esünket, és nem is jogtalanul. Aki egy kicsit is ért a számítógépekhez, az tudja, hogy a 64-es valóban sok mindent tud.

A vásárlás és a kezelési utasítás áttanulmányozása után azonban egy kis csalódást érzünk: egy szó sincs benne a nagyfelbontású grafikáról, nem is beszélve a jelkészlet módosításáról, vagy a megszakítási lehetőségekről. Még a kézikönyv méreteihez képest viszonylag terjedelmes, sprite-okról szóló fejezet sem mond el mindent a készülék valódi értékeiről.

Ne is nézzük túl szigorú szemmel ezt a rövid kis leírást, hisz ha számítógépünk képességeit a lehető legrészletesebben be akarnánk mutatni, hatalmas tankönyvmennyiségre volna szükségünk.

Ennek egy része lehet ez a könyv, ami a 64-es grafikai tudását igyekszik feltárni. Csak megemlítjük, hogy készülékünk 1983-ban elnyerte "Az év számítógépe" megtisztelő címet, amihez minden bizonnyal hozzájárult csodálatos grafikai lehetőségeivel is.

A könyv három jellemző részből áll, azaz a készülék valamennyi jellemzőjét három szempont szerint tárgyaljuk. Ezek:

- hardver
- alapvető programozási ismeretek
- alkalmazás

Első nagyobb fejezetünkben (3. fejezet) mindent megtudunk a VIC (videocontroller) chip képernyővezérléséről, valamint a kép szerkesztésének szabályairól. Megtudjuk, mi a feladata a sok VIC-regiszternek, mik azok a sprite-ok, hogyan kell ezeket programozni, bekapcsolni és kezelni, valamint azt, hogy hogyan kell grafikát készíteni. Tisztázzuk magának a grafikának a fogalmát és azt, hogy mit jelent a jelkészlet vagy a képernyőtároló eltolása. Ezekhez szorosan kapcsolódva megvizsgáljuk a 64-es tárolófelépítését és megnézzük, milyen lehetőségeket kínál. Mindezzel persze semmit sem érünk, míg hiányzik a "know-how", a különböző képernyőfunkciók programozásának tudománya. A 4. fejezetben tehát ezzel foglalkozunk. Megtanuljuk, hogy hogyan kell átállni a grafikus üzemmódra, és megpróbáljuk a legegyszerűbb ábrákat (pont, vonal, kör vagy ellipszis) a képernyőre vinni. Sprite-okat hozunk létre és módosítjuk a jelkészletet. Ehhez két, nagyon kényelmes szerkesztő- (editor-) programot is közlünk, amelyekkel gyorsan és könnyen elvégezhetők a szükséges műveletek. A sprite-ok mozogni fognak, sőt össze is ütköznek megfelelő ellenőrzés és irányítás mellett.

A további fejezetekben megismerkedünk a betöltés, a tárolás, a hardcopy készítés és a megszakítás (interrupt) technikájával. Végül ráadásként átnyújtunk egy kis grafikai programcsomagot, amely hasznos segítséget nyújthat mindazoknak, akik számítógépük ezen kincsét gyorsan, könnyen és mégis eredményesen akarják kiaknázni.

Míg a 4. fejezet a számítógép egyes lehetőségeinek programozásával külön foglalkozik, addig az 5. fejezetben példákat találunk a tanultak alkalmazására és a különböző grafikai eljárások kombinációira. Itt nyílik lehetőségünk az egyes dolgok kölcsönhatásának vizsgálatára.

Reméljük, hogy könyvünk segítségével minden 64-es felhasználó a grafika tökéletes ismerőjévé válik. Olyan szakemberré, aki bármilyen kérdésben tanácsot tud adni, akkor is, amikor mások már nem találnak megoldást. Ha valamit nem tudunk, lapozzuk fel gyorsan a Függeléket és máris megvan a válasz. A könyv végén széleskörű irodalomjegyzék is található a témában eddig megjelent írásokról.

Még valami: Belátjuk, hogy nem mindenkinek van ideje és türelme az összes program, főleg a hosszúak beviteléhez. Ezek nélkül azonban fontos információktól esünk el, ezért készítettünk egy olyan programlemezt, amelyen a könyvben szereplő összes példaprogram megtalálható, előkészítve, működésre készen.

2. FEJEZET

Bitek és byte-ok

A sok különféle tényező rögzítéséhez, amelyekkel a grafikai változatok számtalan lehetőségét meghatározhatjuk, a számítógép tárolóját (memória), illetve az integrált áramkörök különböző regisztereit vesszük igénybe. Azért nem kell megijedni! Szó sincs arról, hogy most bonyolult elektrotechnikai részletekkel kelljen foglalkoznunk. A dolgok megértéséhez elegendő egy kis matematika, ami könnyen áttekinthető és ma már az 5–6. osztályosok tananyagában is szerepel.

Egyebek közt meg kell ismerkednünk a számok kettes (bináris) számrendszerben való ábrázolásával.

Mint ismeretes, a hagyományos számítógépek elektromos vezetékek és építőelemek tömegéből állnak, amelyek összesen két alapvető állapotot érzékelnek: áram van – áram nincs. Ebből egy egészen kicsiny világ jön létre, aminél sokkal többet kívánunk.

Nézzük például a számokat. Hozzárendelhetjük ugyan az "áram nincs" állapothoz a 0-t, az "áram van" állapothoz pedig az 1-est, de ennél jóval több szám létezik, amelyek mindegyikét ábrázolnunk kell.

2.1 A tízes vagy decimális számrendszer

A mindennapi életben tíz számjegy (0–9) áll rendelkezésünkre a számok ábrázolásához. Innen ered az elnevezés is, a latin decem (=tíz) szóból. Ezekből, kis trükk segítségével minden más szám előállítható: nagyobb számok esetén egyszerűen a számjegyek egymás mellé írásával. Ha például 1000-ig akarunk számolni, már 9-nél elfogy a számjegykészletünk. Mindenki tudja, hogy ezután a számolást újra 0-nál kezdjük, de eléje írunk egy 1-est, ezzel jelezve, hogy egyszer már elértük a 9-et. A kilenc után tehát 10 (egy, nulla), 11 (egy, egy), 12 (egy, kettő)... 19 (egy, kilenc) következik. Ekkor

másodszor is elfogy a számjegykészletünk, de ez semmi problémát nem okoz, hiszen ezt az első számjeggyel jelezni tudjuk és a számolást ismét 0-nál kezdjük. Következik tehát a 20 (kettő, nulla), 21 (kettő, egy), 22 (kettő, kettő)... 29 (kettő, kilenc)... stb. Ha az első és második számmal is elérjük a 9-et (99), akkor az első szám elé egy újabb 1-est írunk és újra 0-tól (most már 00) kezdjük a számolást. Így jön létre a 100-as számjegy.

Ezt az elvet követve, tíz számjegyünkkel az összes egész számot ábrázolhatjuk. A számok egyes számjegyei a számban elfoglalt helyük szerint kerülnek megnevezésre: egyes, tízes, százas, ezres... stb. neveken (helyiérték). Így egy szám értéke, melynek számjegyei sorban d_0, d_1, d_2... stb., a következő képlettel számítható ki:

$$Z = d_0 * 10^0 + d_1 * 10^1 + d_2 * 10^2 + ... + d_n * 10^n$$

Megjegyzés:

Bármilyen számot a 0-dik hatványra emelve 1-et kapunk, azaz $5^0=10^0=X^0=1$. Kivétel a 0^0, amely nem meghatározható.

Az ábrázolás egy másik módja a következő (az önkényesen választott szám a 3124):

10^4	10^3	10^2	10^1	10^0
0	3	1	2	4

2.2 A kettes vagy bináris számrendszer

Mint láttuk, összesen tíz számjegyünk van arra, hogy a számokat ábrázoljuk. Szegény számítógépünknek azonban mindössze kettővel (0 és 1) kell megelégednie. Vajon hogy számol 1000-ig?

Egész egyszerűen: úgy, mint mi, vagyis: 0, 1. Itt elfogyott a jelkészlete, tehát a számjegy elé egy 1-est ír és újra 0-tól számol: 10 (egy, nulla), 11 (egy, egy). Ezzel a második helyen is kimerítette a számjegykészletét, tehát megint eléje ír egy 1-est és ismét 0-tól kezdi a számolást: 100 (egy, nulla, nulla).

A számolás tehát így alakul: 00, 01, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011 ... stb.

Mint látjuk, a kettes számrendszerben a tízeshez hasonló elven épülnek fel a számok. Egy bináris szám értékét tehát a következő képlet szerint számíthatjuk ki:

$$Z = b_0 * 2^0 + b_1 * 2^1 + b_2 * 2^2 + \ldots + b_n * 2^n,$$

ahol $b_0, b_1, b_2 \ldots b_n$ az egyes számjegyeket jelölik a legmagasabb bináris helyig. Ha tehát ismerjük egy bináris szám számjegyeit, könnyen kiszámíthatjuk annak decimális megfelelőjét.

Számítsuk át például az 10110100 bináris számot decimálissá:

2^7		2^6		2^5		2^4		2^3		2^2		2^1		2^0
1		0		1		1		0		1		0		0
128	+	0	+	32	+	16	+	0	+	4	+	0	+	0 = 180

A bináris helyet a számítástechnikai szaknyelv bit-nek, információs egységnek nevezi. Egy bitnek két lehetséges állapota van: 0=nem és 1=igen. 8 bit újabb egységet képez, amit byte-nak hívunk. Ezzel a 8 bittel vagy 1 byte-tal 0–255-ig ábrázolhatjuk a számokat. Két byte-tal (16 bit) azonban már jóval több, összesen 65536 szám állítható elő (0–65535).

Ha egy byte decimális értékére vagyunk kíváncsiak, a már ismert képlettel számíthatjuk ki, vagy megfelelő táblázatot is használhatunk.

Az átszámítás fordítva is lehetséges. Alakítsuk vissza a decimális 180-at bináris számmá:

180:128 = 1	maradék	52
52: 64 = 0	maradék	52
52: 32 = 1	maradék	20
20: 16 = 1	maradék	4
4: 8 = 0	maradék	4
4: 4 = 1	maradék	0
0: 2 = 0	maradék	0
0: 1 = 0	maradék	0
azaz $180_{(d)}$	$=10110100_{(b)}$	

Az eljárás a következő: A decimális számot sorban elosztjuk a bináris helyiértékekkel. Az első osztás maradéka a következő osztás osztandója lesz, mindaddig, míg az osztást a legkisebb helyiértékkel (1) is el nem végeztük. A bináris számot az osztások eredményeinek egymás mellé írásával képezzük.

Másik módszer:

180:2=	90	maradék	0
90:2=	45	maradék	0
45:2=	22	maradék	1
22:2=	11	maradék	0
11:2=	5	maradék	1
5:2=	2	maradék	1
2:2=	1	maradék	0
1:2=	0	maradék	1
azaz	$180_{(d)}$	$=10110100_{(b)}$	

Ebben az eljárásban az osztást mindig 2-vel végezzük és a bináris számot a maradékokból képezzük úgy, hogy az utolsó osztás maradéka lesz a legnagyobb, az elsőé a legkisebb helyiértékű szám. Az osztást addig végezzük, míg eredményként 0-t nem kapunk.

A bináris számokat az eléjük írt %-jellel jelölik, amit a továbbiakban mi is alkalmazni fogunk.

Látjuk, hogy a folyamatosan leírt számok végül hosszúak és áttekinthetetlenek lesznek. A számítógép tudja kezelni, de mi jobban szeretjük a régi, bevált decimális írásmódot. A két számrendszer közti átszámítás azonban elég nehézkes, ezért megszületett a tizenhatos számrendszer.

2.3 A tizenhatos vagy hexadecimális számrendszer

Ebben a számrendszerben, mint a neve is mutatja, tizenhat számjegy áll rendelkezésünkre a számok ábrázolásához. Ezek a tízes számrendszer számjegyei (0–9) és az ABC első hat betűje (A–F). A hexadecimális számokat a szám elé írt $-jellel jelöljük.

A hexadecimális számjegyek tehát a következők:

dec.:	0	1	2	3	4	5	6	7	8	9	10	11
hex.:	$0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B
dec.:	12	13	14	15								
hex.:	$C	$D	$E	$F								

A különböző számrendszerek közti átszámítás azonos elven végezhető. Számítsuk át a $FE2A hexa-számot decimálissá:

$16^3=4096$		$16^2=256$		$16^1=16$		$16^0=1$	
F		E		2		A	
15*4096	+	14*256	+	2*16	+	10*1	= **65 066**

Az átszámítás fordítva is lehetséges:

65066:4096=15	(=$F)	maradék	3626
3626: 256=14	(=$E)	maradék	42
42: 16= 2		maradék	10
10: 1=10	(=$A)	maradék	0

$65066_{(d)} = FE2A_{(h)}$

Az eljárás a bináris számok átszámításánál megismert törvényszerűségeket követi.

Milyen előnyei vannak a hexadecimális számrendszernek?

Először is, a számokat röviden és áttekinthetően ábrázolja. Ez azonban még nem döntő. Sokkal lényegesebb az, hogy a hexadecimális–bináris átszámítás nagyon egyszerűen végrehajtható.

Mindig 4 bináris számjegy ábrázol egy hexa-számot. Egy byte-ot tehát 2 hexa-számjeggyel fejezhetünk ki.

Pl.:	%	1101	0110
	$	D	6

Ilyen röviden és szemléletesen írható fel az egyébként hosszú és körülményes bináris szám. A Függelékben találunk egy táblázatot, amely a kettes–tízes–tizenhatos számrendszerek közti átszámításban segítségünkre lehet.

2.4 Logikai műveletek

A byte-ok és tárolórekeszek tartalmának megváltoztatásához és kezeléséhez különböző logikai műveleteket alkalmazunk, amelyekből kettőt röviden ismertetünk: AND és OR. Mindkettő két bináris szám összekapcsolását jelenti.

Itt az ideje, hogy néhány fogalmat tisztázzunk. A következőkben bekapcsolt biten az 1 értékű biteket, a bit bekapcsolásán a 0 értékű bit 1-re váltását,

kikapcsolt vagy törölt biten a 0 értékű biteket, a bitek kikapcsolásán vagy törlésén az 1 értékű bit 0-ra váltását értjük.

És most nézzük a bináris számokkal végzett műveleteket:

a) AND

Tételezzük fel, hogy egy byte-ban csak azokat a bekapcsolt biteket akarjuk meghagyni, amelyek egyidejűleg egy másik byte-ban is bekapcsolt állapotban vannak. Ebben az esetben a két byte-ot AND-del kapcsoljuk össze. A művelet bitről bitre való összehasonlítást jelent és az eredményt a következő táblázat szerint kapjuk meg:

AND	0	1
0	0	0
1	0	1

Ezek szerint az eredmény csak akkor 1, ha mindkét összekapcsolandó bit értéke 1. Minden más esetben az eredmény 0.

Pl.:

```
     10110010

AND  01100111
-------------
     00100010
```

Tehát az első és második byte megfelelő bitjei között egyenként AND-kapcsolatot hoztunk létre.

Ezt a műveletet alkalmazzuk többek között akkor, ha a byte-ban egy bitet úgy akarunk kikapcsolni, hogy a byte többi bitje ne változzon.

b) OR

Az OR-kapcsolatot szintén táblázattal szemléltetjük:

OR	0	1
0	0	1
1	1	1

Eszerint az eredmény akkor 1, ha az összekapcsolandó bitek közül legalább az egyik 1-es.

Pl.:

```
       1011  0010

OR     0110  0111
------------------
       1111  0111
```

Ezt a műveletet akkor alkalmazzuk, ha a byte valamelyik bitjét be akarjuk kapcsolni.

3. FEJEZET

Hardver alapismeretek

A Commodore 64-es rendkívül sok lehetőséget kínál grafikák készítéséhez és vezérléséhez. BASIC-je azonban semmiféle olyan utasítással nem rendelkezik, amellyel ezek a dolgok kihasználhatók lennének. A nagyfelbontású grafikáról a kézikönyv még csak említést sem tesz. Emiatt mindent, ami a grafikával kapcsolatos, közvetlenül gépi nyelven kell megoldanunk. BASIC-ben csak akkor programozhatunk, ha ehhez megfelelő BASIC-bővítővel rendelkezünk. Meg kell azonban mondanunk, hogy semmiféle bővítő nem képes arra, hogy a CBM 64 minden lehetőségére kitérjen. Ezért ahhoz, hogy számítógépünk minden adottságát kihasználhassuk, jól meg kell ismernünk a grafikaszervezés minden fortélyát.

Mielőtt rátérnénk a programozásra és ismertetnénk a felhasználási lehetőségeket, meg kell ismerkednünk az alapokkal: a grafika vezérlésével és szervezésével. Be kell vallanunk, hogy ez a téma nem lesz könnyű. Igyekezni fogunk érthetően és egyszerűen elmagyarázni.

3.1 A VIC regiszterei

Először röviden áttekintjük a VIC (Video Interface Chip) 47 regiszterének rendeltetését, amelyek többek között a teljes képernyővezérlést is ellátják.

Ezek a regiszterek végzik szinte az összes grafikára és szövegre vonatkozó funkció vezérlését. Szerepüket jól meg kell értenünk, mert a képernyőműveletekkel kapcsolatos dolgok mind ezek működésén alapulnak.

Ebben a fejezetben csak rövid áttekintést nyújtunk, de a következőkben mindenre részletesen kitérünk. Ezekkel a dolgokkal munkánk során állandó kapcsolatban leszünk, ezért érdemes kéznél tartani a Függelék erre vonatkozó, regiszterfoglaltsági táblázatát (6.8).

A VIC-regiszter báziscíme: 53248 ($D000).

Reg.-száma dec. hex.	Rövid leírás	Alapállapot dec. bin.
00 $00	A 0. sprite X koordinátája*	00 %0000 0000
	Ezzel a regiszterrel a 0. sprite X koordinátájának 0 és 255 közé eső értékeit ábrázoljuk. Mivel az X koordináta 255-nél nagyobb is lehet, szükségünk van egy további, 9. bitre. Ez az ún. MSB (=legfelső bit), ami nem más, mint a 16-os regiszter 0. bitje.	
01 $01	A 0. sprite Y koordinátája	00 %0000 0000
	Hasonlóan az előzőhöz, de átvitel nélkül.	
02–15 $02–$0F	A többi hét sprite X és Y koordinátái.	
	1. sprite: 2/3 regiszter 3. sprite: 4/5 regiszter . . . stb.	
16 $10	Az X koordináták legfelső bitje	00 %0000 0000
	Ez a regiszter adja az X koordináták ábrázolásához szükséges 9. bitet. Mindegyik sprite-hoz 1 bit tartozik: 0. sprite – 0. bit 1. sprite – 1. bit . . . . . . 7. sprite – 7. bit	

* *Megjegyzés:* A sprite-okat 0–7-ig számozzuk.

17	$11	1. vezérlőregiszter	155	%1001 1011

0–2 bit: képernyőeltolás fel/le

3 bit: =0:24 sor/=1:25 sor

4 bit: =0:képernyő ki/=1:be

A képernyő kikapcsolásakor a VIC nem szakítja meg többé a CPU-t (ami pl. sprite-ok generálásakor 40 millisecundum is lehet). A program futása valamivel gyorsabb és egyenletesebb lesz.

5 bit: =1 – Normál bit-térkép üzemmód

6 bit: =1 – Bővített háttérszín (Extended Colour) üzemmód

7 bit Átvitel a 18-as regiszterből ($12).

18	$12	Rasztersor–IRQ	55	%0011 0111

Ebbe a regiszterbe annak a rasztersornak a számát írjuk, amelynek felépítésekor megszakítást akarunk kiváltani.
Olvasáskor mindig a VIC által éppen létrehozott rasztersor számát tartalmazza. A túlcsordulást a 17-es regiszter 7. bitje veszi át.

19	$13	A fényceruza X koordinátája	00	%0000 0000

Annak a képernyőpozíciónak az X koordinátáját (raszterkoordináta) tárolja, amelyre a fényceruza mutat. Amikor a rasztersorok felépítése éppen ide ér, a fényceruza impulzust küld a számítógépnek, az megállapítja az aktuális képernyő-pozíciót és beállítja a regisztert. (Fényceruza-vezeték=0.)

20	$14	A fényceruza Y koordinátája	00	%0000 0000

Hasonlóan, mint a 19-es regiszter, de az Y koordinátára vonatkoztatva.

21	$15	A sprite-ok be- és kikapcsolása	00	%0000 0000

Itt történik a sprite-ok be- és kikapcsolása. Minden bit egy sprite-ot képvisel (bekapcsolt bit=bekapcsolt sprite)

(ld. a 16-os regiszternél).

22	$16	2. vezérlőregiszter	08	%0000 1000

0–2. bit: képernyőeltolás jobbra/balra

3. bit: =0:38 jel/=1:40 jel soronként

4. bit: =1:többszínű (Multicolor) üzemmód

5–7. bit: nem használt

23	$17	A sprite-ok Y irányú nagyítása	00	%0000 0000

Minden bit egy sprite-ot képvisel. Ha értéke 1, a sprite dupla szélességű lesz (l. még a 29-es regisztert).

24	$18	VIC-báziscímek	20	%0001 0100

Ez a regiszter adja a képernyőtár és a jelkészlettár felső címbitjeit, amelyekkel ezek a tartományok eltolhatók (l. CIA2 0. regisztert is).

0 bit: nem használt

1–3 bit: a jelkészlettár 11–13-as címbitje (*2048)

4–7 bit: a képernyőtár 10–13-as címbitje (*1024)

25	$19	Interrupt Requiest Register (IRR)	15	%0000 1111

Ez a regiszter jegyzi a megszakítás kérésének okát. Minden olvasás után törlődik, ezért a kiolvasott értéket vissza kell írni.

0 bit=1: az aktuális rasztersor száma=a 18. regiszterben tárolt értékkel.

1 bit=1: sprite-háttérütközés (31. reg.)

2 bit=1: sprite–sprite-ütközés (30. reg.)

3 bit=1: fényceruza-impulzus

4–6 bit: nem használt

7 bit=1, ha legalább az első négy bit egyike 1.

26	$1A	Interrupt Mask Regiszter (IMR)	00	%0000 0000

Ebben a regiszterben előírhatjuk, hogy melyik esemény váltson ki megszakítást. A bitek foglaltsága ugyanaz, mint a 25-ös regiszterben. Ha ugyanaz a bit mindkét regiszterben (25/26) bekapcsolt állapotban van, megszakítás jön létre. Más szóval a 26-os regiszter minden bekapcsolt bitje lehetőséget teremt arra, hogy a 25-ös regiszter megszakítást váltson ki.

27	$1B	Elsőbbségi viszony (prioritás)	00	%0000 0000

Minden bit egy sprite-ot képvisel. Ha egy bit értéke 1, a háttérnek, ha 0, a sprite-nak van elsőbbsége. Ez azt jelenti, hogy az első esetben a sprite a háttérjelek mögött, a második esetben pedig azok előtt mozog.

28	$1C	Többszínű sprite-ok	00	%0000 0000

Minden bit egy sprite-ot képvisel. Ha egy bit értéke 1, a hozzá tartozó sprite többszínű üzemmódban jelenik meg.

29	$1D	A sprite-ok X irányú nagyítása	00	%0000 0000

Minden bit egy sprite-ot képvisel. Ha egy bit értéke 1, a megfelelő sprite dupla magasságú lesz.

30	$1E	Sprite–sprite-ütközés	00	%0000 0000

Minden bit egy sprite-ot képvisel. Ha két sprite ütközik, a megfelelő bitek bekapcsolódnak,

		ugyanakkor az ütközést a 25-ös regiszter 2. bitje is rögzíti. Az esemény után ezt a regisztert törölni kell.	
31	$1F	Sprite-háttérjel	00 %0000 0000
		A 30-as regiszterhez hasonlóan, de a sprite-háttérjelütközéseket rögzíti. Egyidejűleg a 25-ös regiszter 1. bitje is bekapcsolódik.	
32	$20	Keretszín	14 %0000 1110
33	$21	0. háttérszín	06 %0000 0110
34–36	$22–$24	1–3. háttérszín	01 02 03
37/38	$25/$26	0/1 sprite többszín regiszterek	04 00
39–46	$27–$2E	Egyszínű sprite-ok 0–7	01 02 03 04 05 06 07

A listában a decimális és hexadecimális számok a regiszterek relatív címeit jelentik, amelyeket vezérléskor a báziscímhez mindig hozzá kell adni. Ezt követi a regiszter neve és alapállapota. Ez utóbbin azt az értéket értjük, amely a számítógép bekapcsolásakor kiindulási értékként a regiszterbe íródik. Ezeket az értékeket binárisan, és BASIC-programokhoz, decimálisan is megadtuk.

A VIC regiszterei mellett létezik még néhány más tárhely is, amelyek különösen a grafikai programozással kapcsolatban érdekesek. A teljesség kedvéért ezeket is felsoroljuk:

a) A sprite-minta mutatója (pointer)

Ahhoz, hogy a VIC számára megadhassuk egy sprite 63 byte hosszúságú mintájának kezdőcímét, a képernyőtár utolsó 8 byte-ját használjuk. Eredeti helyük a 2040–2047 ($07F8–$07FF) tartomány. A sprite-mintákat ún.

blokkokban (lásd később) tároljuk, amelyeket megszámozunk. Ez a szám tulajdonképpen egy mutató, ami azt adja meg, hogy az illető sprite alakja melyik blokkból származzon. (Ha például a 6-os számú sprite-unkat a 14. blokkban tárolt minta szerint akarjuk ábrázolni, akkor az említett utolsó 8 byte közül a hatodikba 14-et írunk. Ez mind a 8 sprite esetében hasonlóan történik.) Ha a mutatót (itt 14) megszorozzuk 64-gyel, megkapjuk a sprite-minta táron belüli relatív kezdőcímét. A mutató értéke 0–255 lehet.

b) A VIC címtartományának legmagasabb értékű címbitjei:

A 24-es VIC-regiszteren kívül létezik még egy további tárrekesz, amellyel pl. képernyőtár vagy a jelkészlettár eltolható. Ez a CIA2 (=Complex Interface Adapter 2) 0. regisztere (=$DD00=56576 rekesz), a 0/1 speciális bitekkel. Ez a két bit adja a VIC-báziscím felső két bitjét (14/15.).

c) Botkomány, potenciométer, billentyűzet:

Itt a CIA 1 első két regisztere az illetékes:

0. regiszter: (cím: 56320=$DC00)

Normál üzemmód:

0–7 bit: a billentyű oszlopának kiválasztása

További feladatok:

0–4 bit:	0-ás botkormány:	0. bit=1: fel
	(1-es kapu=PORT 1)	1. bit=1: le
		2. bit=1: balra
		3. bit=1: jobbra
		4. bit=1: tűz
6–7 bit:	a potenciométer (paddle) kiválasztása	
	(1-es kapu=PORT 1)	6. bit=1: A potenciométer
		7. bit=1: B potenciométer

Egyszerre csak az egyik bit értéke lehet 1!

Botkormány használatakor a 0. regisztert mindenekelőtt bevitelre kell állítani a POKE 56322,224 (2. regiszter=$DC02) utasítással. Visszaállítása a POKE 56322,255-tel történik. (Ez nem érvényes feltétlenül az 1. regiszterre is.)

1. regiszter: (cím: 56321/$DC01)

Normál üzemmód:

0–7 bit: a lenyomott billentyű sorának visszajelzése

További feladatok:

0–4 bit: ugyanaz, mint a 0. regiszter esetében, de 2-es kapura (PORT2) vonatkoztatva (1-es botkormány/potméterek).

Ezzel röviden összefoglaltuk a CBM-64 grafikájához szükséges alapismereteket.

3.2 A VIC üzemmódjai

A VIC sokféle megjelenítési lehetőséget kínál. Ezeket nagyjából három csoportba sorolhatjuk:

- nagyfelbontású grafika – egyedi pont módszer (normál bit-térkép üzemmód),
- sprite-ok,
- szöveg üzemmód (=jelek egy állandó jelkészletből).

Ehhez jön még további kettő, amelyeket az említett három alapüzemmód kiegészítéseként választhatunk:

– egyszínű üzemmód,

– többszínű üzemmód,

valamint a csak szöveg üzemmódban alkalmazható

– bővített háttérszín (extended colour) üzemmód.

Az egyes üzemmódokról itt csak rövid, vázlatos áttekintést nyújtunk, de a későbbi fejezetekben mindegyiket részletesen is ismertetni fogjuk. (A színtár,

képernyőtár, grafikustár, jelkészlettár... stb. kifejezések magyarázatát l. a 3.3 fejezetben.)

A) Nagyfelbontású grafika – egyedi pont módszer

a) *Egyszínű üzemmód*

Ezt az üzemmódot a 17-es VIC-regiszter 5. és 6. bitjének bekapcsolásával választjuk ki. Ezzel egyidejűleg a 22-es VIC-regiszter 4. bitje 0. Lényege az, hogy a képernyő pontjai és a grafikustár bitjei közt közvetlen összefüggés van. A grafikustár minden egyes bitje a képernyő valamelyik pontját képviseli.

A felbontás 320×200 pont, tehát a grafikustár mintegy 8 k memóriaterületet foglal el. A színinformációkat a kb. 1 k nagyságú képernyőtár rögzíti, amelynek minden byte-ja a képernyőn egy-egy 8×8 pontból álló négyzet színét határozza meg.

A grafikustár egy bitje és az általa meghatározott képernyőpont színének származási helye a következő összefüggésben áll egymással:

bit = 0 –	(kikapcsolt pont = háttérszín) a pont színe a képernyőtár alsó négy bitjéből származik.
bit = 1 –	(bekapcsolt pont = pontszín) a pont színe a képernyőtár felső négy bitjéből származik.

b) *Többszínű üzemmód*

Kiválasztása a 17-es VIC-regiszter 5. és 6., valamint a 22-es VIC-regiszter 4. bitjének bekapcsolásával történik.

Ebben az üzemmódban a grafikustár 2–2 pontja határoz meg a képernyőn 1–1 dupla szélességű pontot. A felbontás emiatt csak 160×200 pont, viszont minden 8×8 képernyőpontból álló négyzethez 4 különböző színt rendelhetünk. A színek származási helyét a pontot jelentő két bit együttesen határozza meg, a következők szerint:

bitek: 00 –	a pont (ill. 2 pont) színe a 0. háttérszín-regiszterből származik.
01 –	a szín a képernyőtár alsó négy bitjéből származik.
10 –	a szín a képernyőtár felső négy bitjéből származik.
11 –	a szín a színtárból származik.

B) Sprite-ok

A sprite-ok olyan állandó felbontású, szabadon meghatározható alakzatok, amelyekből a képernyőn egyidejűleg nyolcat jeleníthetünk meg és azokat egymástól függetlenül mozgathatjuk. Ezenkívül változtatható a színük, nagyságuk, a képernyőn elfoglalt helyük, valamint meghatározható az egymással és a háttérjelekkel szembeni elsőbbségi viszonyuk (prioritás). Mozgási tartományuk: 512×256 pont, tehát nagyobb, mint a képernyő. Ezáltal "beúsztathatók" a képbe.

Egy sprite meghatározásához (sprite-minta) 63 byte szükséges.

a) *Egyszínű sprite-ok*

Ebben az üzemmódban a sprite-ok 24×21 pontból építhetők fel. Minden pontot a minta egy bitje jellemez. A színek alakulása:

bit = 0 – a sprite itt átlátszó, vagyis ezen a helyen a háttérjelek látszanak.
bit = 1 – a pont színe a normál sprite-szín-regiszterekből (VIC 39–46) származik.

b) *Többszínű sprite-ok*

Ebben az üzemmódban a minta 2–2 bitje jellemzi a sprite 1–1 dupla szélességű pontját. Emiatt a sprite felépítéséhez csak 12×21 pontunk van, de 4 különböző színt rendelhetünk hozzá. A két bit függvényében a színek származási helye a következő:

bitek: 00 – a sprite itt átlátszó.
01 – a pont színe a 0. többszínregiszterből (VIC 37) származik.
10 – a pont színe az 1. többszínregiszterből (VIC 38) származik.
11 – a pont színe a normál sprite-szín-regiszterekből (VIC 39–46) származik.

A különböző üzemmódban előállított sprite-ok egyidejűleg is megjeleníthetők a képernyőn.

C) *Szöveg üzemmód*

Ebben az üzemmódban egy – a jelkészlettárban (jelgenerátor) rögzített – állandó pontelrendezés kerül, állandó jelként a képernyő valamely 8×8 pontból álló négyzetébe. Eszerint minden jel mintájának rögzítéséhez 8 byte szükséges. A jelkészlettárban található pl. minden betű és szám bitmintája. A képernyőtár egy-egy byte-ja határozza meg, hogy ezek közül (max. 2×256 jel) éppen melyik jelenjen meg a képernyő megfelelő helyén.

a) *Egyszínű (vagy normál) szöveg üzemmód*

Ebben az üzemmódban a képernyőtár byte-jait teljes egészében a jelkészlettár bitmintáira vonatkozó mutatóként használjuk. Ezáltal egyidejűleg összesen 256 különböző jelet ábrázolhatunk a képernyőn. A bitminta egy bitje a képernyőn megjelenő jel 1 pontját képviseli. A színek a következők szerint alakulnak:

bit = 0 – (kikapcsolt pont=háttérszín),
a pont színe a 0. háttérszín-regiszterből származik.
bit = 1 – (bekapcsolt pont=pontszín),
a pont színe a színtár alsó négy bitjéből származik.

b) *Többszínű üzemmód*

Ebben az üzemmódban normál és többszínű jeleket is ábrázolhatunk. Ha a színtár byte-jának 3. bitje 0, a hozzá tartozó jel továbbra is egyszínűként jelenik meg, azzal a korlátozással, hogy a 3. bit más célú foglaltsága miatt most csak 8 színből választhatunk. (16 szín rögzítéséhez 4 bitre van szükség.)

Ha a szóban forgó 3. bit=1, a jel többszínű, azaz a bitminta 2 bitje határozza meg a jel egy dupla szélességű pontját. A felbontás 4×8 pont, viszont minden jelet négy színből állíthatunk össze. A bitek a következő színeket határozzák meg:

bitek: 00 – 0. háttérszín-regiszter (VIC 33).
01 – 1. háttérszín-regiszter (VIC 34).
10 – 2. háttérszín-regiszter (VIC 35).
11 – a színtár alsó három bitje.

A 3-as szín csak az ábrázolható 8 szín valamelyike lehet.
A 0–2 színek minden jelnél azonosak.

c) *Bővített háttérszín-üzemmód*

Ebben az üzemmódban minden jelhez négyféle háttérszínből választhatunk. A jelek normál (8×8) ábrázolásban jelennek meg és a pontok színe (bit=1) a színtárból származik (mint normál esetben). Ezzel szemben a kikapcsolt pontok színei (háttérszínek) különböző forrásból származhatnak, amit a képernyőtár byte-jainak két felső bitje (6. és 7.) határoz meg, a következők szerint:

bitek: 00 – 0. háttérszín-regiszter (VIC 33).
01 – 1. háttérszín-regiszter (VIC 34).
10 – 2. háttérszín-regiszter (VIC 35).
11 – 3. háttérszín-regiszter (VIC 36).

Mivel a képernyőtár 8 bitjéből kettőt a színek meghatározására használtunk fel, a bitminta mutatójaként már csak 6 marad. Ezzel a 6 bittel csupán 64 különböző szám ábrázolható, ezért ebben az üzemmódban egyidejűleg csak 64 különböző jelet vihetünk a képernyőre.

Röviden ennyit a VIC üzemmódjairól, amelyeket a 3.4–3.6 pontokban részletesen is ismertetni fogunk.

3.3 A CBM 64 tárkezelése

E fejezet csak azok számára lesz érthető, akiknek van már némi ismeretük a társzerkezet, tárcímek és hasonlók tekintetében. Akik teljesen laikusok, egyelőre ugorják át. Akik csupán a grafikáról nem tudnak semmit, azok a 3.3.2.1 pont rövid áttanulmányozása után térjenek rá a 3.4 pontra.

Kétségtelen viszont, hogy aki a grafikával hatékonyan akar bánni, annak a következő oldalakat is feltétlenül át kell tanulmányoznia, mert itt mondunk el néhány dolgot a készülék általános tárkezeléséről. Itt kell megjegyeznünk azt is, hogy a legtöbb és legértékesebb funkciók csak gépi nyelven (assembler) érhetők el. Emiatt talán érdemes lenne egyszer ezzel a témakörrel is foglalkozni. Gyakran alaptalanul félnek ettől a nyelvtől, holott azok számára, akik BASIC-ben már programoztak, viszonylag könnyen elsajátítható. Különösen, ha ehhez egy jó könyv is kéznél van, mint pl. az ugyancsak DATA BECKER–NOVOTRADE kiadású "Gépi kódú programozás a Commodore 64-esen", vagy "Gépi kódú programozás haladóknak"

című könyvek. Aki megismerkedik vele, hamarosan szívesebben programoz majd assembler-ben, mint a lassú BASIC-ben.

Aki mégis a BASIC-et választja, annak különböző grafikus bővítőkre van szüksége, amelyek szintén léteznek a 64-esre. Ezekben olyan egyszerű parancsok állnak rendelkezésünkre, amelyekkel a programozás szinte gyerekjáték.

Érdemes figyelemmel kísérni a piacot. Ne felejtsük: grafikánál nagyon fontos tényező a gyorsaság!

Érdemes sok mindent kipróbálni, hogy mindig a legmegfelelőbb megoldást választhassuk.

De térjünk a lényegre:

A teljes képernyőtartalmat tárolnunk kell ahhoz, hogy a VIC mindig tudja, mit kell a képernyőre vinnie. Éppen ezért akár szöveg, akár grafika tárolásáról van szó, viszonylag nagy tártartományra van szükség. A grafikus tár például mintegy 8 k, a hozzá tartozó szín 1 k, sőt többszínű üzemmódban 2×1 k tárkapacitást igényel. Ez a tárterület egyéb célokra nem használható, de a munka optimalizálása érdekében lehetőségünk van arra, hogy táron belüli elhelyezkedését megválaszthassuk. Ugyanakkor olyan területek is vannak, amelyeket egyáltalán nem, vagy csak nagyon nehezen tudunk használni (pl. BASIC-ből).

A következők megértéséhez kénytelenek vagyunk néhány szót szólni számítógépünk tárfelépítéséről:

A Commodore 64-es legfőbb eleme a 6510-es mikroprocesszor, amelyet CPU-nak (Central Processing Unit)-nak is nevezünk. Ez a számítógép minden folyamatában részt vesz. Ugyanazzal az utasításkészlettel rendelkezik, mint elődje, a 6502-es, ezért azzal szoftverkompatíbilis. A 6510-es néhány további kivezetéssel rendelkezik, amelyek többek között a tárkezelésben is szerepet játszanak. Két saját regisztere van, a 0-ás és 1-es. Számunkra csupán az utóbbinak van jelentősége.

A másik, amiben megegyezik a 6502-sel, az ún. címtartomány. Ezen azt a tárnagyságot értjük, amit a számítógép vezérelni tud. Esetünkben ez 64 k, mert a címzéshez, tehát a vezérléshez, összesen 16 bit áll rendelkezésre, amelyekkel a 0–65535 ($0000–$FFFF) címek érhetők el.

Azt már tudjuk, hogy a CBM 64-es önmagában 64 k RAM-tartománnyal rendelkezik. (RAM alatt értjük az írható és olvasható – Random Access Memory – tárterületet. Ennek tartalma a készülék kikapcsolásakor elvész. Úgy

is mondhatjuk, hogy ez a számítógép munkatárja.) Ehhez jön még a ROM-tartomány, ami a RAM-mal ellentétben csak olvasható (Read Only Memory). Tartalma kikapcsolás után is megmarad. Feladata az operációs rendszer, a BASIC-értelmező, a jelgenerátor... stb. tárolása. Ezenkívül a különböző egyéb építőelemek (VIC, CIA, SID...) ROM-tartományai és regiszterei.

Ezek együttesen még 24 k-t foglalnak el.

De mi legyen ennyi tárterülettel, ami ráadásul normál módon nem is vezérelhető?

A megoldás az ún. tárátfedés. Ennek az a lényege, hogy több tártartomány (pl. ROM és RAM) ugyanazokkal a címekkel rendelkezik, tehát ugyanazt a tárterületet foglalja le.

Alapállapotban (a gép bekapcsolásakor) ez így néz ki:

$0000... $7000	$8000 $9000	$A000 $B000	$C000	$D000	$E000 $F000
R		A		M	
	esetleges modultartomány (ROM)	Basic-ROM		jelgenerátor I/O-tartomány	Kernal-ROM

Bekapcsolás után azok a tartományok olvashatók, amelyek a rajzon alul láthatók, tehát:

$0000–$9FFF: RAM
$A000–$BFFF: ROM
$C000–$CFFF: RAM
$D000–$DFFF: I/O
$E000–$FFFF: ROM

Ha valamilyen modult (cartridge) is használunk, az a $8000–$9FFF tartományba kerül.

A számítógéppel közölni kell, hogy melyik tartományt vezérelje. Ez több tényezőtől is függ. A legfontosabbak:

- hozzáférés írásra vagy olvasásra,
- a tárt a VIC vagy a 6510-es akarja-e használni,
- kell-e vezérelni a $D000–$DFFF tartományt, vagy nem,
- az 1. regiszter 0–2 bitjeinek tartalma.

Alapvetően különbséget kell tennünk a VIC és a 6510-es között, mivel ezek gyakorlatilag ugyanabban az időben különböző tártartományokat vezérelnek.

Ha a vezérlést a VIC végzi, úgy kell eljárnunk, mint a grafikánál. Az egyes tárakat (pl.: képernyőtár) olyan tárterületeken kell elhelyeznünk, amit a VIC elér.

A 6510-es vezérlése akkor is közvetlenül érint bennünket, ha grafikával, ill. képernyővezérléssel nincs dolgunk. BASIC-ben meglehetősen korlátozottak a lehetőségeink, mert sok tartomány szinte állandó használatban van, gépi nyelven (assebler) viszont mindennel szabadon rendelkezhetünk.

3.3.1 A 6510-es tárkezelése

3.3.1.1 Egy byte olvasása

Ha a tár valamelyik byte-ját ki akarjuk olvasni (pl. PEEK-kel, vagy ugyanezt jelenti egy gépi nyelvű program lefuttatása is), szükség lehet az egymást átfedő tárterületek közti átkapcsolásra. Az éppen hozzáférhető tartomány a CPU ún. adatregiszterének (1. regiszter) első három (0–2) bitjétől függ. A készülék bekapcsolása után mindhárom bit értéke 1, és az első fejezetben vázolt tárfelosztást eredményezi.

A) *0/1 bit - LORAM/HIRAM*

a) *0/1 bitek = 11*

Ha mindkét bit értéke 1, a $A000–$BFFF címeken a BASIC-ROM, a $E000–$FFFF címeken pedig a Kernal-ROM vezérelhető. Ez az alapállapot. Modul beiktatásakor ehhez jön még a $8000–$9FFF modul-ROM vezérlése. Az elrendezés vázlata a következő:

$0000...	$7000	$8000 $9000	$A000 $B000	$C000	$D000	$E000 $F000
R	A	M esetleges modul-ROM	Basic-ROM		I/O	Kernal-ROM

b) *0/1 bitek = 10*

Ha az 1. bit értéke 1, akkor a modul- és BASIC-ROM helyett az alatta levő RAM-terület olvasható ki; az alábbi elrendezés szerint:

$0000...	$7000	$8000	$9000	$A000	$B000	$C000	$D000	$E000	$F000
R		A			M		I/O	Kernal-ROM	

Ebben az elrendezésben azt is megtehetjük (BASIC-ből is), hogy pl. a BASIC-értelmezőt az alatta levő RAM-ba másoljuk, ahol némi változtatás után módosított BASIC-ként szerepelhet. Nézzünk erre egy példaprogramot:

```
1 REM *** P 1 ***
2 REM
10 FOR AD=10*1049 TO 12*4096-1
20 POKE AD,PEEK(X):REM BASIC-ROM MASOLASA A RAM-BA
30 NEXT AD
40 REM A ROM-TARTALOM MODOSITHATO
50 POKE 1,PEEK(1) AND 254:REM RAM BEKAPCSOLASA
```

Ügyeljünk arra, hogy a modul-üzemmódban a $8000–$9FFF tartományt is át kell másolni az alatta levő RAM-ba, mert ilyenkor a CPU a modul-ROM-hoz sem fér hozzá.

Az 1. regiszterrel mindig nagyon óvatosan kell bánni, mert minden változtatásnak komoly következményei vannak a tár felépítésében. Hibás foglaltság esetén gyakran csak a gép ki- és bekapcsolásával segíthetünk magunkon. (Ne féljünk, ettől szerencsére semmi baja sem lesz!)

c) *0/1 bitek = 01*

Ha az 1. bitet kapcsoljuk ki és csak a 0. bit értéke 1, a teljes ROM-tartomány hozzáférhetetlenné válik. Ez az I/O-tartományt nem érinti, az továbbra is ugyanúgy vezérelhető. Ebben az esetben összesen 60 k elérhető RAM-területtel rendelkezünk.

$0000...	$7000	$8000	$9000	$A000	$B000	$C000	$D000	$E000	$F000
R		A				M	I/O	R A M	

d) *0/1 bitek = 00*

Ha mindkét bitet töröljük, a teljes RAM-terület rendelkezésünkre áll. A ROM-ok és az I/O-elemek viszont többé nem olvashatók ki (kivéve a VIC-et és más IC-ket, mert ezekhez továbbra is hozzáférnek, ezért a kép megmarad!). Az elrendezés a lehető legegyszerűbb:

$0000...	$7000	$8000	$9000	$A000	$B000	$C000	$D000	$E000	$F000
R				A				M	

Ez az egyetlen lehetőségünk arra, hogy az I/O és a jelgenerátor alatti 4 k RAM-hoz hozzáférjünk.

B) *2. bit - CHAREN*

Ez a bit arra alkalmas, hogy az eltolható jelkészlettárat a 6510-es (azaz a programozható) számára olvashatóvá és ezáltal másolhatóvá és módosíthatóvá tegye.

a) *2. bit = 1*

Ez a bit csupán a tár $D000–$DFFF tartományára van hatással, a többi területet nem befolyásolja. Alapállapotban értéke 1. Ezzel a 4 k-t lefoglaló jelkészlettár, ami éppen ebben a tartományban helyezkedik el, csak a VIC által olvasható. A CPU ehelyett itt az I/O-címekkel, a VIC, SID (Sound Interface Device) és a CIA-k regisztereivel, továbbá a színtárral rendelkezik. Nézzük az elrendezés vázlatát:

$0000 ...	$D400 ...	$D800 ...	$DC00	$DD00	$DE00	$DF00
VIC-reg.	SID-Reg	szín-RAM	CIA1	CIA2	I/O 0	I/O 1

b) *2. bit = 0*

A 2. bit törlésével a CPU (ill. a programozó) számára kiolvashatóvá válik a jelkészlettár tartalma. Ebben az esetben ez a teljes I/O-tartományt (4 k) lefoglalja.

Ha ezzel BASIC-ben próbálkozunk, a számítógép a következő 1/60 másodpercben elbúcsúzik tőlünk és kezelhetetlenné válik. Ennek az az oka, hogy normál körülmények között minden 1/60 másodpercben lefut az ún. megszakító- (interrupt-) rutin (ld. 3.7 fejezet), amit a CIA 1-ben található időzítő irányít. Miután a 2. bit törlésével a CIA 1-et kiiktatjuk, többé a megszakítórutin sem hívható le. Gépi nyelvű programozáskor a SEI-utasítással először be kell kapcsolnunk az ún. megszakításjelzőt (interrupt-flag), hogy a rutin lehívását megakadályozzuk. Később ez a CLI-utasítással oldható fel. Ugyanezt kell tennünk, ha az 1. bit törlésével kiiktatjuk a Kernal-ROM-ot, mert maga a megszakítórutin itt található.

3.3.1.2 Egy byte írása

Alapvetően mások a körülmények, amikor a tárba (pl. POKE-kal) írni akarunk.

Mivel a ROM-területek átírása nem célszerű, mindenféle írási kísérlettel csak az alattuk levő RAM-ot érhetjük el, függetlenül a CPU 1. regiszterének tartalmától. Ez alól csak egyetlenegy kivétel van: a $D000–$DFFF tartomány. Itt normál esetben csak az I/O-területet érhetjük el. Ha az itt található RAM-ba is írni akarunk, két lehetőségünk van:

- minden ROM-ot kikapcsolunk,
- bekapcsoljuk a jelkészlettárat.

Az első esetben a CPU 1. regiszterének 0. és 1., az utóbbiban pedig a 2. bitjét kell kikapcsolnunk.

Ezek után már az előző kis program (P1) is tökéletesen érthető. A 20. sorban PEEK (AD)-vel kiolvassuk a BASIC-ROM tartalmát, amit ugyanerre a címre, de most már a RAM-ba, POKE-kal visszaírunk.

3.3.2 A VIC tárkezelése

A CPU mellet a VIC-nek (videocontroller) is hozzá kell férnie a tárhoz, hogy a képernyőre vonatkozó, szín- és pontelrendezéssel kapcsolatos információkat felhasználhassa. Ebben az esetben nincs értelme annak, hogy akár a Kernal-ROM, akár a BASIC-értelmező elérhető legyen, ezért a CPU-hoz hasonlóan lehetőség van az egymást átfedő tárterületek közti átkapcsolásra. Ezt a VIC

önállóan végzi. Hasonlóan sok lehetőségünk van, mint a CPU-nál. Csupán azt kell tudnunk, hogy a VIC melyik címtartományokat vezérli (és milyen céllal) és melyiket nem. Ugyanakkor más szempontból a VIC tárkezelése rendkívül dinamikus. Bizonyos feltételek betartása mellett ugyanis önkényesen kijelölhetjük a VIC egyes tárainak helyét. Más szóval, lehetőségünk van arra, hogy a grafikus tárat, jelkészlettárat, képernyőtárat és a sprite-blokkokat a táron belül tetszőlegesen helyezzük el.

Mielőtt erre rátérnénk, néhány szót kell ejtenünk a VIC-kel kapcsolatos tárolási feladatokról, és tisztáznunk kell azokat a fogalmakat, amelyekkel a munkánk során nap mint nap találkozni fogunk.

3.3.2.1 A VIC-kel kapcsolatos tárolási feladatok

A VIC, különféle feladatai (szöveg, grafika, sprite-ok, szín, ...) végrehajtásához, viszonylag nagy tárterületet igényel. Ezek:

- jelkészlettár,
- képernyőtár (video-RAM),
- színtár (szín-RAM),
- grafikustár,
- sprite-blokkok.

Ezekkel már korábbi fejezeteinkben is találkoztunk, a következőkben pedig részletesen kifejtjük, mit is jelentenek és hogyan használjuk ezeket. Feladatukat és pontos felépítésüket a 3.4–3.6 fejezetekben ismertetjük. Az itt található felsorolás csupán tájékoztató jellegű és a fogalmak meghatározása a célja.

a) *Jelkészlettár*

Jelkészlettár (jelgenerátor) alatt azt a tártartományt értjük, amely az egyes képernyőjelek alakját, az ún. bitmintát tárolja és lehetővé teszi, hogy egy billentyű lenyomásakor a megfelelő jel megjelenjen a képernyőn.

Összesen 2×2 k tárkapacitással rendelkezik és 512 jelre vonatkozó információt tartalmaz. Ezeknek csak egy része jeleníthető meg egyidejűleg a képernyőn (l. 3.2, 3.6, 4.4 fejezetek).

b) *Képernyőtár*

A képernyőtár kb. 1 k tárkapacitást köt le és különböző feladatai vannak. Alapállapotban (a készülék bekapcsolása után) jeltárként működik, ami nem tévesztendő össze a "jelkészlettár"-ral. Az éppen képernyőn levő jelek ún. képernyőkódjait tartalmazza, amik az ASCII-kódhoz hasonlóak és a jelkészletre vonatkozó mutatóként működnek.

Grafikus üzemmódban a képernyőtárnak színtároló feladata van. Innen származnak a pont- és háttérszínek, valamint többszínű üzemmódban a grafikus képernyő minden 8×8 (ill. 4×8) pontjának 1-es és 2-es színei (ld.: 3.4, 3.6, 4.2, 4.4 fejezetek).

A képernyőtár utolsó 8 byte-nek feladata a sprite-mutatók tárolása.

c) *Színtár (szín-RAM)*

A színtár a képernyőtárhoz hasonlóan kb. 1 k tárkapacitást foglal le a $D800–$DBFF (55296–56319) címtartományban. Byte-jai olvasással és írással is hozzáférhetők, de csak az alsó négy bitjük használható. A felső négy bit nem módosítható, értékük mindig 1. Szöveg üzemmódban a képernyőjelek színét tárolja.

Grafikus üzemmódban csak többszínű állapotban van jelentősége. Innen származik a grafikus képernyő minden 4×8 pontjának 3. színe.

d) *Grafikustár*

Ez, mint a neve is mutatja, a grafikus képet tárolja. A grafikus képernyő minden egyes pontját külön rögzíti, ezért nagyfelbontású grafika esetén 320×200 = 64 000 pontot tárol, pontonként 1 bittel. Ebből könnyen kiszámítható, hogy kb. 8 k tárkapacitást köt le.

e) *Sprite-minták*

A sprite-ok alakjának rögzítésére különböző tárterületeket foglalhatunk le, egyenként 63 byte-tal. Ebben a 63 byte-ban a normál (egyszínű) sprite 24×21 pontját egyenként határozzuk meg.

Többszínű üzemmódban szintén 63 byte-ra van szükség sprite-onként, azzal a különbséggel, hogy két-két bit adja a sprite egy dupla szélességű pontját.

3.3.2.2 A VIC tárvezérlése

A VIC esetében sokkal egyszerűbb és áttekinthetőbb a különböző tártartományok vezérlése, mint a CPU-nál. Abból az egyszerű alapelvből indulhatunk ki, hogy ha más előírás nincs, a VIC hozzáférhet a RAM-hoz, még a $D000–$DFFF (533248–57343) címtartományban is. Ez még akkor is így van, ha a programozó csak a ROM-területeket érheti el. Független tehát attól, hogy mi van a CPU 1. regiszterében. Ez különösen fontos ezért, mert így megoldható, hogy pl. a grafikustárat a ROM alatti RAM-területre helyezzük, ezáltal BASIC-tárterületeket takaríthatunk meg. Ezt az ismert SUPER-GRAPHIK 64 is megvalósítja.

Ez az alapelv a következőkre érvényes:

- képernyőtár,
- grafikustár,
- sprite-blokkok,
- jelkészlettár.

Ez a négy tár tehát mindig a RAM-ból származik – két kivétellel:

Ha egy beállítással (ld. következő fejezet) a $1000–$1FFF (4096–8191) vagy a $9000–$9FFF (36864–40960) területet akarjuk használni (pl. a grafikustárat a $8000–$9FFF tartományba helyezzük, vagy a 64–127-es sprite-blokkok számára a $1000–$1FFF területet választjuk ki), az információk nem a RAM-ból, hanem a jelkészlet-ROM-ból származnak, ami a $D000–$DFFF címtartományban van.

Ez a különlegesség abból ered, hogy a számítógép bekapcsolása után a VIC a 64-es tárterületéből csak az alsó 16 k-t tudja vezérelni. A jelkészlet, ami

köztudottan a képernyőjelek megjelenítéséhez szükséges, az I/O-címek alatti $D000–$DFFF területet foglalja le. Emiatt a $1000–$1FFF tartománynak különleges szerep jutott, a VIC számára ugyanis itt található a jelkészlet. Az egyéb funkciók ilyenkor sajnos áldozatul esnek. A $9000-es tartomány is e különleges helyzet eredménye.

Ezzel kapcsolatban meg kell jegyeznünk, hogy a különböző funkciók közvetlen címzésekor (tehát nem közvetve a $1000 címen keresztül) a programozó a $D000 tartományban nem a jelkészlet-ROM-ot, hanem az alatta levő RAM-ot vezérli.

A tárfelosztás vázlatosan a következő:

$0000	$1000	$2000 ...	$9000	$A000	$B000	$C000	$D000	$E000	$F000
R A M	üres	R A M	üres	R			A	M	
							jelkészlet-ROM		

Ennyit csupán a teljesség kedvéért. Bővebbet a következő fejezetből tudhatunk meg.

A színtár esetében a dolog talán még egyszerűbb: mivel ennek, eltérően a többi tartománytól, nem változtatható a táron belüli elhelyezkedése, mindig a $D800–$DBFF (55296–56319) címtartományban található. Csupán arra kell ügyelnünk, hogy ez egy külön, az alatta levő RAM-tól különböző területet foglalja el, ami a programozó számára az I/O-címek síkjában van. A színtár $D800-as címét tehát nem szabad összetéveszteni a normál RAM $D800-as címével.

3.3.2.3 A VIC-tárak áthelyezése

A VIC tárkezelésében az egyik legszebb és legpraktikusabb dolog az, hogy az egyes tárak a teljes tárterületen belül eltolhatók. Megtehetjük pl. azt, hogy a grafikus tárat, ami a grafikánkat rögzíti, a $2000 (8192) címre helyezzük, de ha ezzel zavarjuk a BASIC-tartományunkat, akár a $E000 címre is eltolhatjuk, a ROM alá. Alkalmazhatunk két vagy több grafikus oldalt, így míg az egyiken dolgozunk, a másik látható a képernyőn, ahogy erre a SUPERGRAPHIC 64 is módot ad. Eltolhatjuk a jelkészlettárat is és elkészíthetjük saját jelkészletünket, ami mindenféle különleges jeleket tartalmazhat (pl. ékezetes betűk). Ezzel a 3.6 és 4.4 fejezetekben még találkozni fogunk.

Bármilyen táráthelyezésről legyen is szó, mindig tartsuk szem előtt, hogy a VIC melyik tartományokat képes vezérelni. Ezt az előző fejezetben már ismertettük. Így pl. a sprite-blokkok sosem kerülhetnek a \$1000–\$1FFF területre, mert a VIC ezeken a címeken nem a RAM-ot, hanem a \$D000–\$DFFF tartományban elhelyezkedő jelkészlet-ROM-ot éri el.

A következő fejezetek tanulmányozásához rendkívül fontos, hogy a 3.3.2.2 fejezetet tökéletesen megértsük.

a) *Általános táráthelyezés*

A VIC önmagában csak 16 k-t képes címezni (\$0000–\$3FFF vagy %0000 0000 0000 0000–%0011 1111 1111 1111). A mi címtartományunk viszont összesen 64 k, azaz éppen négyszer akkora. A VIC két felső (14. és 15.) címbitje hiányzik. Ezt kívülről kell megoldani. Itt az a regiszter az illetékes, amelyet már a 3.1 fejezetben is említettünk, vagyis a

CIA2 0. regisztere: \$DD00 (=56576), 0. és 1. bitek.

Ez a két bit adja a VIC hiányzó címbitjeit, a címbitek között aláhúzva:

címbitek: \$ F E D C B A 9 8 7 6 5 4 3 2 1 0

Ha ezeket egyszerűen beállíthatnánk, máris meglenne a teljes címünk. Van azonban egy probléma. Mindkét bit alacsony-aktív, ami azt jelenti, hogy bekapcsoláskor törölt bitnek számít és fordítva. Emiatt ahhoz, hogy a helyes címet kapjuk, először invertálnunk kell ezeket. Ezután automatikusan minden, VIC által vezérelhető tártartomány(kivéve a színtárat) 16 k-s lépésenként eltolható.

A következő táblázatban összefoglaltuk, hogy a VIC a két felső bit állapotától függően mely tárterületeket vezérli:

Bitek 0/1	Címbitek	Vezérelhető címtartomány
11	00	\$0000–\$3FFF (0–16383)
10	01	\$4000–\$7FFF (16384–32767)
01	10	\$8000–\$BFFF (32768–49151)
00	11	\$C000–\$FFFF (49152–65535)

A 0/1 bitek a CIA 2 0. regiszterének bitjeit, a címbitek pedig a 14 és 15. bitet jelentik. Az eredeti állapot: 0/1=11, tehát a táblázat első sora. Csak így lehet a képernyőtár a $0400–$07FF (1024–2047) között és a jelkészlet-ROM (a $1000 cím különleges helyzetéből adódóan) a $D000 (53248) címen.

Egy példa: Tételezzük fel, hogy valamilyen okból át akarjuk helyezni a képernyőtárat a $C400 (50176 = 49152+4*256) címre. (Tekintsünk most el attól, hogy további változtatások nélkül a képernyőn levő jelek többé nem módosíthatók.) Ehhez csupán a következő programsorra van szükség:

```
POKE 56576, PEEK (56576) AND 253 OR 0
```

a VIC máris a $C000–$FFFF tartományból veszi az információkat, vad káoszt okozva a képernyőn. Egyedül a színek módosíthatók rendesen, mivel a színtár nem helyezhető át.

Az eredeti állapotot a

```
POKE 56576, PEEK (56576) AND 253 OR 3
```

programsorral állíthatjuk vissza. Ha ezt a billentyűzetről visszük be, vigyázzunk, hogy ne kövessünk el hibát és ne felejtsük el lenyomni a RETURN-t!

b) *A képernyőtár áthelyezése*

Az általános táreltoláson kívül a 16 k-s címtartományokon belül a képernyőtár kisebb lépésekben, külön is eltolható. Ebben a 24-es VIC-regiszter 4–7 bitjei játszanak szerepet (ld. a 3.1 fejezet). Ez a 4 bit adja a képernyőtár 10–13. címbitjeit a következő ábra szerint.

Címbitek	$F E	D C B A	9 8 7 6 5 4 3 2 1 0
	CIA 2 0. reg. 0/1 bitek	VIC 24. reg. 4–7 bitek	VIC – intern

A képernyőtárat tehát a 16 k-s címtartományon belül 1 k-s lépésenként tolhatjuk el. A készülék bekapcsolása után a szóban forgó négy bit értéke: %0001, a képernyőtár $400 (1024)-től kezdődik és szövegtárként működik. Fontos megjegyezni, hogy ez a cím – eltérően az a) alpontban leírtaktól – csak és kizárólag a képernyőtárra érvényes. Ha csak ezeket a biteket módosítjuk,

a többi tár a helyén marad. Egy példa: a képernyőtárat a $400 (1024) címről a $800 (2048)-ra akarjuk eltolni, ami persze meglehetősen képtelen dolog, hiszen itt kezdődik a BASIC-tár. Mindenesetre a következő programsorra van szükségünk:

```
POKE 53248+24, PEEK (53248+24) AND 15 OR 2*16
```

A visszatérés a

```
POKE 53248+24, PEEK (53248+24) AND 15 OR *16
```

utasítássorral végezhető.

Erre akkor lehet szükségünk, ha pl. olyan hosszú a BASIC-programunk, hogy a $400–$7FF tártartományt is fel kell használnunk, vagy két szövegoldallal dolgozunk stb.

c) *A jelkészlettár áthelyezése*

A képernyőtáron kívül a kiválasztott 16 k-s címtartományban a jelkészlettár is eltolható. Ebből a szempontból ismét a 24-es VIC-regiszter a meghatározó, ami néhány címbit közbenső tárjaként működik. Ezúttal csak három bitet használhatunk, amelyek a 11–13 címbiteket állítják elő. Emiatt az összes 2*2 k nagyságú jelkészlettárat csak 2 k-s lépésenként tudjuk eltolni:

Címbitek	$F E	D C B	A 9 8 7 6 5 4 3 2 1 0
	CIA 2 0. reg. 0/1 bitek	VIC 24. reg. 1–3 bitek	VIC – intern

Érdekes, hogy a számítógép operációs rendszere is ezt a három bitet használja. Ha a SHIFT/C= billentyűkombinációval átkapcsolunk az alternatív jelkészletre, a számítógép a jelkészletcím 11. bitjét a 24-es VIC-regiszter 1. bitjének módosításával változtatja meg (ld. 3.6 fejezet).

Nagyon fontos, amit a 3.3.2.2 fejezetben a $1000–$1FFF tartomány különleges helyzetéről mondtunk. Ha a jelkészlettár áthelyezésére vállalkozunk, pl. azért, hogy saját jelkészletünket használhassuk, legjobb, ha újból elolvassuk a 3.3.1 fejezetet is.

d) *A grafikustár áthelyezése*

A grafikustár helyét is megválaszthatjuk. Mivel ez 8 k, csak kétszer fér el a 16 k-s címtartományban. Ebből eredően egy bit is elegendő az áthelyezéshez (természetesen az általános eltoláson kívül). Ez a 24-es VIC-regiszter 3. bitje, aminek így gyakorlatilag két feladata van: a jelkészlet eltolása és a grafikustár eltolása.

Ugyanúgy, mint a jelkészlet esetében, a grafikustárnál is a 13. címbitet adja:

Címbitek	$F E	D	C B A 9 8 7 6 5 4 3 2 1 0
	CIA 2 0. reg. 0/1 bitek	VIC 24. 3. bit	

A $0000–$3FFF (0–16383) VIC-címtartományban például a grafikustárat a $0000, vagy a $2000 (8192) címtől kezdődően helyezhetjük el. (Ilyenkor a második lehetőséget célszerű választani, mert ellenkező esetben tönkretesszük a nulláslapot, a veremtárakat ... stb.

A téma folytatásaként térjünk rá a 3.4 fejezetre!

3.4 A pontgrafika

A regiszterek és a tárhasználat részletes ismertetése után foglalkozzunk a nagyfelbontású, ill. a többszínű grafikával, vagyis az ún. pontgrafikával. Nevét onnan kapta, hogy minden pontja külön vezérelhető. Ez rendkívül érdekes téma és elengedhetetlen mindazoknak, akik grafika programozásával foglalkoznak. Ez és a következő négy fejezet azokat a hardver-szintű előfeltételeket ismerteti, amik a 4. fejezetben összefoglalt grafikai lehetőségek kihasználásához szükségesek. Mindenképpen foglalkozzunk ezzel a témakörrel, még akkor is, ha nem értünk mindent. Legalább a grafika felépítésével tisztában kell lennünk. Látni fogjuk, hogy a készülék színelőállítása meglehetősen bonyolult, de azért ne adjuk fel! Egy ilyen könyvet legalább kétszer el kell olvasni és néhány fejezettel naponta foglalkozni kell, ha valóban érteni akarunk a grafikához.

3.4.1 A színek

A Commodore 64-es mind szöveg, mind pedig grafikus üzemmódban 16 színt tud előállítani és használni. Ezek mindegyike ún. színkóddal rendelkezik, amit a jelek előállításában részt vevő regiszterekben bináris számként tárolunk. Ha pl. a 32-es VIC-regiszterbe POKE-utasítással 0-t írunk, a képernyőkeret színe fekete lesz. A következőkben felsoroljuk a 64-es által előállítható színeket és a hozzájuk tartozó színkódokat (a Függelékben egy részletesebb táblázat is található):

Kód		
Dec.	Hex.	Szín
0	$00	fekete
1	$01	fehér
2	$02	piros
3	$03	türkiz
4	$04	ibolya
5	$05	zöld
6	$06	kék
7	$07	sárga
8	$08	narancs
9	$09	barna
10	$0A	világospiros
11	$0B	szürke 1
12	$0C	szürke 2
13	$0D	világoszöld
14	$0E	világoskék
15	$0F	szürke 3

Ez a táblázat a grafikában állandó kísérőnk lesz. A színek alkalmazásával külön fejezetekben foglalkozunk.

3.4.2 A nagyfelbontású (NF) grafika

Számítógépünk kétféle grafikus üzemmódot ismer:

- nagyfelbontású (NF) grafika,
- többszínű (TSZ) grafika.

A nagyfelbontású grafikus képernyő vízszintesen (x irány) 320, függőlegesen (y irány) 200 pontból áll. Ez összesen 64000 pontot jelent a képernyőn, egyenletes elosztásban.

A grafikát ugyanúgy tárolni kell, mint a szöveget vagy a színeket. Ahhoz, hogy a képernyőn vagy monitoron megjelenő képet folyamatosan láthassuk, a VIC-nek azt kb. 1/20 másodpercenként újra meg újra elő kell állítania (ld. a fényceruzával foglalkozó fejezetet is).

Pontgrafika esetében a képet az ún. grafikustárban rögzítjük, ahol a képernyő minden pontját egy bit képviseli. A bitek, vagyis a képernyő pontjai, külön-külön vezérelhetők. A 2. fejezetből már tudjuk, hogy a bit egy információs egység, ami kétféle értéket vehet fel: 0-t vagy 1-et. 8 egymást követő bit jelent 1 byte-ot, amit jelnek vagy szónak is nevezünk. Egy byte-nak $2^8=256$ különböző értéke lehet, amit a bekapcsolt (1) és kikapcsolt (0) bitek különböző kombinációi adnak.

Egy byte tehát a képernyő 8 pontját képviseli, ami azt jelenti, hogy a 64000 pont információit 64000/8 = 8000 byte-ban (kb. 8 k) tudjuk tárolni. A 3.3.2.3 fejezetből tudjuk, hogy ezt a 8 k-t a 64 k-os tártartományon belül bárhol elhelyezhetjük.

Nézzük meg, hogyan épül fel a grafikus kép a tárolt információkból.

a) *A grafika felépítése*

Az előbbiek alapján azt gondolhatnánk, hogy a grafikustár a képernyő közvetlen leképezése, azaz pontról pontra követik egymást a megfelelő bitek és byte-ok. Nos, a dolog mégsem ilyen egyszerű. A grafikustár ugyanolyan felépítésű (hardver-szinten egyszerűbben megoldva), mint a később ismertetésre kerülő jelkészlettár. Egy, a képernyőn megjelenő jel, mint tudjuk, 8×8 pontból áll. Ugyanilyen 8×8 mezőkből épül fel a grafikustár is, ahol minden ilyen mezőt a tár 8–8 byte-ja alkot.

Egy ilyen mező tehát 8 pont szélességű és 8 pont magasságú, ezért vízszintesen 320/8 = 40, függőlegesen pedig 200/8 = 25 fér el belőle a képernyőn (ugyanúgy, mint szöveg üzemmódban).

Egy byte egy ilyen mező egy sorának (8 pont) információit hordozza. 8 byte egymás alatt (ill. a tárban egymás mellett) alkot egy blokkot. Az összetartozó 8 byte azonos helyiértékű bitjei a képernyőn 8 egymás alatti pontot jelentenek. A következő ábra segítségével mindez talán érthetőbb lesz:

		0. oszlop	1. oszlop	....
	Bit:	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	stb.
0. sor	0. byte	. . . .	. . . .	
	1. byte	. . . .	. . . .	
	2. byte	. . . .	. . . .	
	3. byte	. . . .		
	4. byte	. . . .		
	5. byte	. . . .		
	6. byte	. . . .		
	7. byte	. . . .		
1. sor	320. byte	. . . .	. . . .	
	321. byte	. . . .	. . . .	
	322. byte	. . . .		
	stb.			

A generálás tehát sorról sorra ezekből a blokkokból történik. Nézzük a kifejezéseket: oszlopnak mindig a 8 pont szélességű függőleges "sávot" nevezzük, amiből a képernyőn egymás mellett pontosan 40 fér el. A sor ugyanez vízszintesen és ebből 25 fér el egymás alatt. Minden oszlop 8 függőleges és minden sor 8 vízszintes pontsorból áll, így a képernyő éppen 320 függőleges és 200 vízszintes pontsorra bontható. Minden oszlop és minden sor 0-tól kezdődően számozva van, így az egyes pontok helyzete könnyen meghatározható. Ezeket a megállapításokat mindig tartsuk fejben a félreértések elkerülése végett.

Az egyes pontok vezérléséhez célszerű volna azok helyzetét valamilyen könnyen kezelhető módon meghatározni. Az előbb már említettük, hogy a képernyőn minden vízszintes és függőleges pontsornak sorszáma van. Legegyszerűbb tehát, ha megadjuk a pont ún. koordinátáit, vagyis hogy hányadik pontként szerepel a vízszintes (x) és a függőleges (y) pontsorban. Ilyen módon a képernyő valamennyi pontja egyértelműen meghatározható. A ponthoz tartozó byte és bit tárbeli helyét a 4.2 fejezetben található képlettel számíthatjuk ki. (Ehhez, a sorok kezdőcímeinek megadásával, a Függelékben levő táblázat is segítségünkre van.)

Végül egy megjegyzés: Már bizonyára észrevettük, hogy az említett 8 k

tárterületnek nem használjuk fel minden byte-ját, csak pontosan 8000-et ($1F40). A maradék 192 byte ($C0) más célokra használható. A későbbiekben látni fogjuk, hogy ugyanez a többi tártartományra is vonatkozik.

b) *Színfelépítés*

Minden nagyfelbontású képhez, azaz minden grafikustárhoz tartozik egy színtár is. Ez nem más, mint a már más vonatkozásban megismert képernyőtár. Alapállapotban szöveg vagy általános jel tárolására alkalmas. Pl. egy billentyű lenyomásakor a VIC innen veszi a képernyőn megjelenítendő jelre vonatkozó információkat. Ebből az következik, hogy ha grafikus üzemmódban nem toljuk el más tártartományba, hanem ott hagyjuk, ahol a szöveget tároljuk, akkor a grafikus képernyőn kis színes négyzet formájában láthatók lesznek az eredeti szöveg képernyőjelei.

(A képernyőtár áthelyezését ld. a 3.3.2.3 fejezetben!)

Az elmondottakat a P.2 programmal szemléltetjük:

```
1 REM *** P 2 ***
2 REM
100 V=53248:REM VIC KEZDOCIME
110 REM
120 REM ***********
130 REM * 1. RESZ *
140 REM ***********
150 REM
160 REM GRAFIKA BEKAPCSOLASA:
170 POKE V+17,PEEK(V+17) OR 6*16
175 REM VIC17 5. ES 6. BIT BEKAPCSOLASA
180 REM GRAFIKUS TAROLO $2000-RE:
190 POKE V+24,PEEK(V+24) OR 8:REM VIC24 3.BIT BE
200 REM WAIT 198,255:GOTO 220:REM VARAKOZAS
210 END
220 REM
230 REM ***********
240 REM * 2. RESZ *
250 REM ***********
260 REM
270 REM GRAFIKA KIKAPCSOLASA:
280 POKE V+17,PEEK(V+17) AND 9*16+15
285 REM VIC17 5. ES 6. BIT KIKAPCSOLASA
290 REM JELGENERATOR VISSZAALLITASA:
300 POKE V+24,PEEK(V+24) AND 15*16+7
305 REM VIC24 3.BIT KIKAPCSOLASA
310 END
```

Ez a program kétféleképpen működhet: egyrészt változtatás nélkül, a lista szerint, másrészt úgy, hogy a 200. sorban az első REM-et elhagyjuk.

Az első esetben a program az átkapcsolás után rögtön befejeződik és a gép visszatér közvetlen üzemmódba. Ha most bármit beviszünk a billentyűzetről, az nem normál módon, hanem kis színes négyzetként fog megjelenni a képernyőn. Az eredeti szöveg-üzemmódot RUN220-szal állíthatjuk vissza.

A második esetben a program a grafikus üzemmód bekapcsolása után egy billentyű lenyomására vár. Bármelyik billentyű lenyomásával automatikusan kikapcsolódik a grafika és a gép újból szöveg-üzemmódban van (ld. a 4.2 fejezetet is!).

Ez a példa a grafika és szín összefüggését szemlélteti. Meggyőződhetünk arról, hogy a képernyőtár egy byte-ja a grafikus képernyő 8×8-as pontmezőjének színét határozza meg. Minden pontmező két színt vehet fel. Egyik az ún. háttérszín, amit a kikapcsolt (0) bitek, másik a pontszín, amit a bekapcsolt (1) bitek adnak. Mindkét esetben a lehetséges 16 szín közül választhatunk. A képernyőtár byte-jainak alsó négy bitje adja a háttérszínt, a felső négy bitje pedig a pontszínt. A színfelbontás így sokkal kisebb, mint amit a grafika valójában megengedne, de egyrészt 64000 pont egyedi színvezérlése túl sok tárhelyet igényelne (összesen 64000*4 = 256000 bit = 32 k), másrészt a hagyományos színes tv-k úgysem tudnák ezt megfelelően megvalósítani. Ráadásul olyan jelentősen csökkenne a feldolgozási sebesség, hogy emiatt külön grafikus processzort kellene beiktatni.

A képernyőtár felépítése valamivel egyszerűbb, mint a grafikustáré. Itt a megfelelő byte-ok és sorok egymás után következnek:

Sor/Oszlop	0	1	2	3	4	5	6	7	...	39
0	$00	01	02	03	04	05	06	07	...	1D
1	$1E	1F	20	21	22	23	24	25	...	4F
2	$50	51	52	53	.	.	.			
.	.									
.	.									
24	$3C0	3C1	...							

Nagyfelbontású grafikánál a képernyőtár minden byte-ja egy 8×8-as pontmező színét határozza meg. Ha tehát ismerjük egy pont relatív tárcímét (leszámítva a grafikustár kezdőcímét), azt 8-cal elosztva megkapjuk a képernyőtár hozzá tartozó byte-jának relatív címét. Ehhez a képernyőtár kezdőcímét hozzáadva kapjuk a byte abszolút tárcímét (gondoljunk utána!).

3.4.3 A többszínű (TSZ) grafika

Mielőtt továbbmennénk, nézzük át még egyszer az előző fejezetet, mert az ott elmondottakat itt már ismertként kezeljük.

Mint tudjuk, a Commodore 64-es nagy grafikus felbontóképességgel rendelkezik, amihez viszont nem társul nagy színfelbontás. E probléma enyhítésére a konstruktőrök létrehoztak egy másik grafikus üzemmódot, amelyet nagyobb színfelbontás, de kisebb grafikus felbontás jellemez: ez az ún. többszínű (multicolor) üzemmód.

A többszínű üzemmód egy-egy 8×8-as pontmező számára kettő helyett négy szín egyidejű alkalmazását teszi lehetővé. Ebben az esetben is a 8 k nagyságú grafikustárat használjuk, de most ennek minden két bitje határoz meg a képernyőn egy dupla szélességű pontot. A felbontás ezek szerint úgy alakul, hogy vízszintes irányban 160 dupla szélességű, függőleges irányban pedig továbbra is 200, egyszeres magasságú pont lesz (a felbontás: 160×200 pont). A színek ezúttal nem csak a képernyőtárból származnak, hanem kiegészítésként a VIC 0. háttérszín-regiszterét és a színtárat is használjuk. Ezeket a területeket négy, ún. színcsatornára osztjuk és 0-tól 3-ig számozzuk (%00–%11). Minden bitpár, ami egy pont előállításáért felelős, felveheti a 0–3 értékeket, meghatározva ezzel a VIC számára az aktuális színcsatornát. (Nagyfelbontású üzemmódban ez úgy történt, hogy a pontot jelentő bit- vagy a háttérszín-csatornát (0), vagy a pontszíncsatornát (1) jelölte ki.) A grafikustárban tehát nem maga a szín található, hanem az, hogy a pont színére vonatkozó információ, a színkód, hol van.

A bitpár–csatorna–tár közti összefüggést a következő táblázat szemlélteti:

Bitkód	Csatorna	Tár
00	0	0. háttérszín-regiszter (VIC 33)
01	1	képernyőtár alsó négy bitje
10	2	képernyőtár felső négy bitje
11	3	színtár

Bitkód alatt azt a bináris számot értjük, amit az adott pontért felelős bitpár ábrázol. A csatornához tartozó tártartomány a tár azon része, amelyet a csatorna, ill. a bitkód adott beállítása aktivizál.

Egy példa: a grafikustárban egy pontot a 01 bitkóddal ábrázolunk. Ez az 1-es színcsatornán keresztül a képernyőtár alsó négy bitjében rögzített színt írja

elő a pont számára. Az adott byte címét ugyanúgy állapítjuk meg, mind NF üzemmódban (3.4.2 fejezet).

Itt még az az új, hogy a 3-as szín, ill. színcsatorna a színtárból származik. Ez alapállapotban a képernyőtárban rögzített szöveg színéért felelős. Nem tolható el a táron belül, ezért többszínű és szöveg üzemmód egyidejű alkalmazásakor kényszerű átfedések fordulnak elő. A színtár felépítése ugyanolyan, mint a képernyőtáré, ezért az itt levő byte-ok címe a már ismert módon határozható meg (ld. 3.4.2 fejezet).

Többszínű üzemmódban is 8×8 (ill. a dupla szélességű pontok miatt 4×8) pont színét tudjuk a képernyőtár és a színtár byte-jaiban rögzíteni. Ezenkívül minden 8×8 ponthoz egyedi színkombinációt rendelhetünk, csupán a háttérszín származik az egész grafikára vonatkozóan ugyanabból a byte-ból (VIC 33 regiszter).

Többszínű grafikák készítésekor ügyelnünk kell az x irányú torzításra.

Nézzük, hogy alakul a grafikus tároló felépítése:

		0. oszlop				1. oszlop				....
	Bit:	7 6	5 4	3 2	1 0	7 6	5 4	3 2	1 0	stb.
0. sor	0. byte	<->	<->	<->	<->	<->	<->	<->	<->	
	1. byte	<->	<->	<->	<->	<->	<->	<->	<->	
	2. byte	<->	<->	<->	<->	<->	<->			
	3. byte	<->	<->	<->	<->					
	4. byte	<->	<->	<->	<->					
	5. byte	<->	<->	<->	<->					
	6. byte	<->	<->	<->	<->					
	7. byte	<->	<->	<->	<->					
1. sor	320. byte	<->	<->	<->	<->	<->	<->	<->	<->	
	321. byte	<->	<->	<->	<->	<->	<->			
	322. byte	<->	<->	<->	<->					
	stb.									

A "↔"-jel az összetartozó biteket jelöli.

Itt hívjuk fel még egyszer a figyelmet arra, hogy a nagyfelbontású és többszínű üzemmódban nem ugyanaz a képernyőtár feladata.

Van még néhány dolog, ami ezzel a témakörrel kapcsolatos, de ezekre majd a 4. fejezetben térünk ki.

3.5 A sprite-ok

A Commodore 64 egyik legérdekesebb tulajdonsága, hogy sprite-okat tud előállítani és kezelni. A sprite-ok olyan kicsiny grafikák, amelyek mind a grafikus-, mind szövegképernyőn megjeleníthetők és egymástól, valamint az egyéb képernyőtartalomtól függetlenül mozgathatók. Egyszerre 8 sprite lehet a képernyőn.

Meghatározhatjuk színüket, nagyságukat, valamint az egymással és a háttérjelekkel szembeni elsőbbségi viszonyukat (prioritás). Lekérdezhetjük a sprite–sprite és a sprite–háttérjel-ütközéseket.

A grafikához hasonlóan ezeknek is két fajtájuk van:

- egyszínű,
- többszínű.

Ezt minden sprite-ra külön megválaszthatjuk.

Minden sprite-okra jellemző funkciót a VIC különböző regiszterei vezérelnek.

Mindenekelőtt ismerkedjünk meg a sprite-ok felépítésével. Érdemes átolvasni még egyszer a 2. és 3.1 fejezetet, mert bizonyos mértékig bináris műveletekkel fogunk dolgozni. A következő két dolgot állandóan észben kell tartanunk:

1. A képernyőn egyszerre max. 8 sprite jeleníthető meg.
2. Minden sprite-nak száma van (0–7).

3.5.1 Az egyszínű sprite-ok felépítése és színezése

Az egyszínű sprite 504 pontból áll, amelyek mindegyike külön be- és kikapcsolható. Ez az 504 pont egy 24×21-es pontmezőt alkot, azaz egy sprite 24 pont széles és 21 pont magas.

Definiálásához, azaz a sprite-minta tárolásához, 504/8 = 63 byte szükséges, mivel minden egyes pontot a tároló egy bitje képvisel. A sprite-ok szélessége 24 pont, ami azt jelenti, hogy egy sort 3 byte alkot. (Az első sort tehát az első három byte, a másodikat a 3–5 byte, ... stb.) Ezt szemlélteti a következő rajz:

		0. oszlop								1. oszlop								2. oszlop							
	sor/bit:	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0. byte	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1	3. byte	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
2	6. byte	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
3	9. byte	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
4	12. byte	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
										stb.															
20	60. byte	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Egy sprite meghatározása tehát azt jelenti, hogy egymás után, sorról sorra és oszlopról oszlopra tároljuk a megfelelő byte-okat. Ennek BASIC-változatára a 4. fejezetben találunk példát.

A képernyőn megjeleníthető 8 sprite-hoz külön-külön a rendelkezésünkre álló 16 szín bármelyikét hozzárendelhetjük. A pontok színe a normál sprite-színregiszterekből (VIC 39–46) származik. Minden sprite-ot egy ilyen regiszter jellemez. Ha pl. az 5-ös számú sprite-ot fehér színben akarjuk ábrázolni, akkor az illetékes VIC-regiszter a 39+5 = 44-es lesz, a $D02C (53292) tárcímre 1-et (a fehér szín kódja) kell beírni.

A kikapcsolt pontok "átlátszóak", ezért ezeken a helyeken a háttérjelek látszanak.

3.5.2 A többszínű sprite-ok felépítése

Nem minden sprite áll 24×21 pontból. A többszínű sprite-ok a többszínű grafikához hasonlóan épülnek fel, azaz a vízszintes irányú felbontásuk felére csökken, miután minden pontjuk dupla szélességű lesz. Ennek eredményeként viszont minden sprite-hoz a háttérszínnel együtt négy színt rendelhetünk, szemben a nagyfelbontású üzemmóddal, ami csak 2 színt (háttérszín és

pontszín) enged meg. Ez a négy szín a grafika négy színcsatornájához hasonlítható és különböző VIC-regiszterekből származik.

A többszínű sprite minden egyes pontjára jellemző két bit (bitkód) adja az aktuális színcsatorna számát (0–3), ugyanúgy, mint a többszínű grafikus üzemmódban. A bitkódok és színcsatornák összefüggése:

Bitkód	Színcsatorna	Színtár
00	0	átlátszó, vagyis a háttérjelek látszanak
01	1	0. többszínregiszter (VIC 37)
10	2	1. többszínregiszter (VIC 38)
11	3	normál sprite-színregiszterek (VIC 39–46)

Látjuk, hogy a 3-as csatornából származó szín sprite-onként különböző lehet, mert ebben a tekintetben minden sprite külön regiszterrel rendelkezik. Az 1. és 2. szín, ill. a háttérszín azonban mindegyiknél azonos, mert ezek ugyanazokból a regiszterekből származnak.

Ellentétben a CBM 64 kézikönyvében leírtakkal, a többszínű sprite-ok programozásakor mind a 16 szín felhasználható.

Azt, hogy a sprite a képernyőn többszínűként jelenjen-e meg, a 28-as VIC-regiszter ($1C) vezérli. Ennek minden bitje egy sprite-ot jellemez. A bekapcsolt bitek (1) jelentik a többszínű, a kikapcsoltak (0) pedig az egyszínű sprite-okat.

A bitek és sprite-ok kapcsolatát a következő táblázat szemlélteti:

Bit:	b7	b6	b5	b4	b3	b2	b1	b0
Értéke:	128	64	32	16	8	4	2	1
Sprite:	s7	s6	s5	s4	s3	s2	s1	s0

Ha pl. a 4-es számú sprite-ot többszínűként akarjuk ábrázolni, akkor a 28-as VIC-regiszter 4. bitjét be kell kapcsolnunk. BASIC-ben ez a következő utasítássorral történik:

```
POKE 53248+28, 16
```

Több sprite (pl. 1-es, 5-ös, 7-es) esetében az utasítássor így alakul:

```
POKE 53248+28, 1 OR 32 OR 128
```

vagy

```
POKE 53248, 1+32+128
```

Az OR-művelet ebben az esetben kivételesen összeadással is helyettesíthető.

A többszínű sprite felépítése teljesen megegyezik az egyszínűével. A bitminta tárolása ebben az esetben is 63 byte-ot igényel, amit a következő ábra szemléltet:

		0. oszlop				1. oszlop				2. oszlop			
sor/bit	:	7 6	5 4	3 2	1 0	7 6	5 4	3 2	1 0	7 6	5 4	3 2	1 0
0	0. byte	<->	<->	<->	<->	<->	<->	<->	<->	<->	<->	<->	<->
1	3. byte	<->	<->	<->	<->	<->	<->	<->	<->	<->	<->	<->	<->
2	6. byte	<->	<->	<->	<->	<->	<->	<->	<->	<->	<->	<->	<->
3	9. byte	<->	<->	<->	<->	<->	<->	<->	<->	<->	<->	<->	<->
4	12. byte	<->	<->	<->	<->	<->	<->	<->	<->	<->	<->	<->	<->
					.	stb.							
20	60. byte	<->	<->	<->	<->	<->	<->	<->	<->	<->	<->	<->	<->

A ”↔”-jel itt is az egy pontot jelentő, összetartozó biteket jelenti.

3.3.3 A sprite-ok alakja – színek

Mielőtt egy sprite-ot képernyőre viszünk, van még néhány dolgunk.

Először is el kell képzelnünk az alakját és azt a hozzá tartozó 63 byte-ban rögzítenünk kell. Ehhez némi segítséget és trükköt találunk a 4.3 fejezetben.

Ezután el kell döntenünk, hogy ezt a 63 byte-ot a tárban hol helyezzük el (ld. a 3.3.2 fejezetet). A tárban alapállapotban (a készülék bekapcsolása után) csak 4 különböző sprite meghatározására van hely, ezért ha ennél többet akarunk rögzíteni, fel kell emelnünk a BASIC-tár kezdőcímét. Ez eredetileg \$0801=2049, ami mutatóként a 43/44-es tárrekeszekben található. Ezt kell POKE-utasítással az új kezdőcímnek megfelelően módosítani. Ne felejtsük el, hogy ezt még a BASIC-program betöltése előtt el kell végeznünk, mert ha ezután módosítjuk a BASIC-kezdőcímet, az értelmező nem fogja megtalálni a programunkat. Másik lehetőség az, hogy áthelyezzük a képernyőtárat,

felszabadítva ezzel az általa elfoglalt tárterületet ($400–$7FF). Ha egyidejűleg szövegkivitelt is végzünk, a képernyőtár kezdőcímének felső byte-ját a 648-as ($288) rekeszbe POKE-utasítással be kell írnunk. (A felső byte értéke alapállapotban: $04. Kiszámítása decimálisan: FB=INT (képernyőtár kezdőcíme/256). Ezt kell a 648-as rekeszbe írnunk.)

Egy példa: Tételezzük fel, hogy a képernyőtárat áthelyeztük a $0800 (=2048)-as kezdőcímre. (Ez a BASIC-kezdőcím áthelyezését is feltételezi.) Adjuk meg az értelmező számára (szöveg kiviteléhez) az új kezdőcímet:

```
POKE 648, INT (2048/256)
```

Ezután feltétlenül törölnünk kell a képernyőt (SHIFT/CLR-HOME, vagy PRINT CHR$(147).

A legtöbb esetben azonban elegendő négy sprite-ot rögzíteni, mert mint látni fogjuk, az azonos alakúakat nem kell külön-külön tárolni.

Ahhoz, hogy a VIC megtalálhassa a sprite-mintát (az adott 63 byte-nyi információt), 16 k-s címtartományát 64 byte-onként blokkokra kell osztani. Összesen 256 ilyen blokkot hozhatunk létre, amelyeket 0–255-ig megszámozunk. Alapállapotban a blokkok címei a következők:

Blokk száma	Kezdőcím
0	$0000–0
1	$0040–64
2	$0080–128
3	$00C0–192
4	$0100–256
.	.
.	.
255	$3FC0–16320

A VIC-címtartomány áthelyezésekor (CIA 2, 0/1 bitek) a fenti kezdőcímekhez hozzá kell adni az új báziscímet.

Az ilyen blokkban 1 sprite bitmintája tárolható. Az utolsó byte-nak nincs jelentősége, mert ehhez csak 63 byte szükséges.

A gép bekapcsolása után az alábbi blokkok állnak rendelkezésünkre:

Blokk száma	Címtartomány
11	$02C0–$2FE (704–766)
13	$0340–$037E (832–894)
14	$0380–$03BE (896–958)
15	$03C0–$03FE (960–1022)

Meg kell jegyeznünk, hogy az utolsó három tartomány egyben a szalagpuffer is (828–10199), ezért Datasette használatakor törlődik. Ilyenkor a sprite-ok csak a 128. blokktól ($2000=8192) kezdődően tárolhatók, mivel a $1000–$1FFF (4096–8192) tartomány, különleges helyzete miatt (ld. 3.1 fejezet), erre a célra nem használható. (Hosszú BASIC-programoknál, vagy nagy tárigény esetén vigyázzunk!) A 16 k-os címtartomány áthelyezésekor további korlátozások is előfordulhatnak.

Mit kezdjünk mindezzel?

Nagyon egyszerű. Ahhoz, hogy a VIC megtalálja a kérdéses sprite-mintát, a képernyőtár utolsó 8 byte-jában (ld. 3.1 fejezet) rögzítenünk kell az azt tároló blokk számát. (Alapállapotban a képernyőtár utolsó nyolc byte-jának címe: $07F8–$07FF (2040–2047)).

Ha utánaszámolunk, hogy a szövegképernyő tárolásához hány byte-ra van szükségünk, pontosan 1000-et (25×40) kapunk. Ezzel szemben a képernyőtár 1 k, azaz 1024 byte-tal rendelkezik. A maradék 24 byte-ra általában nincs szükség, mert az utolsó 8 byte kivételével, ahol a blokkszámokat rögzítjük, szabadon felhasználhatók.

A byte-ok és blokkszámok kapcsolatát a következő táblázat szemlélteti:

Byte:	$07F8 2040	07F9 2041	07FA 2042	07FB 2043	07FC 2044	07FD 2045	07FE 2046	07FF 2047
A sprite száma	0	1	2	3	4	5	6	7

A megadott címek a bekapcsolás utáni alapállapotra vonatkoznak és a képernyőtár áthelyezésével természetesen megváltoznak.

Egy példa: Tételezzük fel, hogy egy sprite-mintát a $03C0–$03FE (960–122) tartományban, azaz a 15. blokkban rögzítettünk. Azt akarjuk, hogy a 2-es és a 6-os sprite-unk eszerint épüljön fel. Az alábbi utasítássorra van szükségünk:

```
POKE 2040+2,15 : POKE 2040+6,15
```

A 15-ös értéket, mint a 15-ös blokk mutatóját, beírjuk a sprite-ok számának megfelelő regiszterbe. Láthatjuk, hogy több sprite alakja ugyanaz lehet, azáltal, hogy a nekik megfelelő byte-ba ugyanazt a mutatót írjuk.

Ha a 14-es blokkban egy másik sprite-minta található és azt szeretnénk, hogy a 2-es sprite ilyenre változzon, csak az alábbi utasítássort kell bevinnünk:

```
POKE 2040+2,14
```

(ld. 4.3 fejezet).

A fenti két sprite-ot vigyük a képernyőre.

Ehhez először is be kell kapcsolnunk ezeket. Ezt a feladatot a VIC 21-es regisztere (sprite ki/be) látja el, amelyben (hasonlóan a 28-ashoz) minden bit egy sprite-ot képvisel.

Bit:	b7	b6	b5	b4	b3	b2	b1	b0
Értéke:	128	64	32	16	8	4	2	1
Sprite:	s7	s6	s5	s4	s3	s2	s1	s0

A 2-es és 6-os sprite-ot tehát az alábbi utasítássorral kapcsolhatjuk be:

```
POKE 53248+21, 64 OR 4
```

vagy

```
POKE 53248+21, 64+4
```

Ezzel még valószínűleg nem fognak megjelenni a képernyőn, mert általában úgy kell beúsztatni. Ilyenkor, bár be vannak kapcsolva, a képernyőhatárokon kívül helyezkednek el. A sprite-ok programozásáról bővebbet a 4. fejezetből tudhatunk meg.

3.5.4 A sprite-ok egyéb tulajdonságai

A definiálással, színkiválasztással és bekapcsolással még nem tudunk mindent a sprite-okról. A következőkben azokkal a tulajdonságokkal foglalkozunk, amik a sprite-ot SPRITE-tá teszik.

3.5.4.1 A sprite-ok elhelyezése a képernyőn (pozicionálás)

A sprite-ok legelső és legfontosabb jellemzője, hogy képernyőn elfoglalt helyük egyenként előírható. Ez nagyon fontos, mert ezáltal hozható létre a mozgás és egyéb különleges hatások (ld. 4. fejezet).

A képernyőt ún. X és Y koordinátákkal jellemezzük úgy, ahogy azt a grafikánál is tettük.

Erre vonatkozóan a következő szabályok érvényesek.

A koordináták mindig a sprite bal alsó sarkának helyét jelölik ki. A sprite-ok mozgástartománya 512×256 pont, azaz nagyobb, mint a képernyő.A képernyő felbontása megegyezik a nagyfelbontású grafikáéval, tehát vízszintes irányban 320, függőlegesen 200 pontból áll. Ez teszi lehetővé, hogy a sprite-okat folyamatosan kiúsztathassuk a képből úgy, hogy a képernyő szélén a sprite egyre csökkenő része látható. (Ugyanez fordítva is igaz.)

A sprite-ok koordináta-rendszerének nulla pontja jóval kijjebb esik, mint a képernyő bal felső sarka. Az első látható pont koordinátái ebben a rendszerben: X=20, Y=30. Ez egyben a normál grafika nulla pontja. Itt a sprite még teljes egészében láthatatlan.

Ha a sprite-koordinátákat grafikus koordinátákká akarjuk átszámítani, az X értékből 20-at, az Y-ból pedig 30-at kell levonni. A fordított átszámítás értelemszerűen a koordináták növelésével történik. Az X = 320+20 és Y = 200+30 pontokban a sprite már nem látható.

A sprite-ok helyzetét az első 17 VIC-regiszterben (0–16) tároljuk. A 16-os regiszternek különleges szerepe van, amire később még visszatérünk. A többit párosával rendelték hozzá a sprite-okhoz. A regiszterpár első tagja az X, második az Y koordinátákat tárolja:

Sprite:	s0	s1	s2	s3	s4	s5	s6	s7
X reg.:	0	2	4	6	8	10	12	14
Y reg.:	1	3	5	7	9	11	13	15

Ha a 6-os sprite-ot az X=100, Y=150 pontba akarjuk helyezni, a következő utasítássorral tehetjük meg:

```
POKE 53248+2*6, 100
POKE 53248+2*6+1,150
```

Mint tudjuk, egy byte max. 256 különböző értéket vehet fel. A sprite-ok mozgásakor azonban X irányban ennél nagyobb értékekre is szükség van, egészen 512-ig. Emiatt igénybe kell vennünk egy ún. segédregisztert. Ez az a bizonyos 16-os regiszter, amiben minden bit egy sprite-ot képvisel és 9. bitként a 256-nál nagyobb értékek ábrázolását teszi lehetővé. (Ez az ún. MSB = Most Significant Bit = legmagasabb értékű bit.) Nézzük a bitek és sprite-ok kapcsolatát:

Bit:	b7	b6	b5	b4	b3	b2	b1	b0
Értéke:	128	64	32	16	8	4	2	1
Sprite:	s7	s6	s5	s4	s3	s2	s1	s0

Amikor tehát 255-nél nagyobb X koordinátát akarunk meghatározni, be kell kapcsolnunk a 16-os regiszter megfelelő bitjét. Ezzel az X koordinátaregiszterben levő értékhez 256-ot adunk.

3.5.4.2 A sprite-ok nagyítása

Elkészült a sprite-unk, elégedettek is lehetnénk, de számítógépünk további meglepetéseket tartogat számunkra. A VIC regiszterei közül kettőt még meg sem említettünk. Ez a 23-as és 29-es. Közreműködésükkel a sprite-unkat nagyíthatjuk.

A 23-as felel az X irányú, a 29-es az Y irányú nagyításért. A nagyítási tényező mindkét irányban 2, azaz a sprite minden pontja kétszeres szélességű és magasságú lehet. A nagyítás két irányban külön is és együtt is megvalósítható.

A két regiszter működése az előzőekhez hasonló. Minden sprite-hoz 1 bit tartozik, amelyet ha bekapcsolunk, létrejön a nagyítás. Lehetőségeink:

Nagyítás:	Nagyítási tényező:	Pontmező:
nincs	1×1	24×21
X irányú:	2×1	48×21
Y irányú:	1×2	24×42
X/Y irányú:	2×2	48×42

Pontmezőn itt képernyőpontot értünk. A sprite-mintában továbbra is az eredeti (24×21) bit szerepel. A 48×42 pont tehát a sprite által maximálisan befedhető képernyőrész. Megjegyzendő, hogy a nagyított sprite-ot sosem

tudjuk teljesen kiúsztatni a képernyőből, még akkor sem, ha a sprite-koordináták nullák.

3.5.4.3 Elsőbbségi viszony (prioritás)

Mi történik akkor, ha két sprite, vagy egy sprite és egy háttérjel a képernyő azonos pontján jelenik meg? Fedjék-e egymást, és ha igen, melyik legyen "elöl"? Ebben a fejezetben erről lesz szó.

a) *Sprite–sprite-átfedés*

Ez az eset viszonylag egyszerű. A sprite-ok, mint tudjuk, sorszámmal rendelkeznek. Ha két sprite a képernyő ugyanazon pontján jelenik meg, akkor az fogja fedni a másikat (az lesz elöl), amelyiknek a sorszáma kisebb. Pl. a 0-ás és 5-ös sprite találkozásakor a 0-ás fogja takarni az 5-öst, természetesen csak azokban a pontokban, amelyek bekapcsoltak.

b) *Sprite-háttérjelátfedés*

Ez az eset valamivel komplikáltabb, igaz, érdekesebb is. A továbbiakban háttérjelen a sprite-on kívüli egyéb képernyőtartalmat értjük, ami lehet betű, különleges jel vagy grafika.

Mint már említettük, meghatározhatjuk, hogy egy sprite a háttérjelek előtt vagy mögött jelenjen-e meg. Ezt a feladatot a VIC 27-es regisztere látja el. Felépítése már ismerős, minden bitje egy sprite-ot képvisel. (Hasonlóan a 16, 21, 23, 28 és 29 regiszterekhez.) A kikapcsolt bitekhez tartozó sprite-ok a háttérjelek mögött, a bekapcsoltakhoz tartozók pedig azok előtt fognak megjelenni.

Ezek birtokában háromdimenziós grafikákat és mozgásokat is előállíthatunk. Erről a 4. fejezetből még többet megtudunk.

3.5.4.4 Ütköztetés

A sprite-ok egymás közti és a háttérjelekkel való ütköztetésének és ezek regisztrálásának főleg a játékokban van jelentősége. Ebben a VIC 30-as és 31-es regiszterei játszanak szerepet. Felépítésük az eddigiekhez hasonló, vagyis minden bitjük egy sprite-ot képvisel.

A 30-as regiszter automatikusan rögzíti a sprite–sprite-ütközéseket oly módon, hogy a részt vevő sprite-okhoz tartozó bitek bekapcsolódnak. A 6-os és 2-es sprite ütközésekor tehát a regiszter tartalma: %0100 0010 lesz. Ez mindaddig a tárban marad, míg azt a programozó a

```
POKE 53248+30,0
```

utasítással ki nem törli.

Ezzel a két bittel egyidejűleg egy másik bit is bekapcsolódik, méghozzá a 25-ös VIC-regiszter 2. bitje. Ez jelenleg még teljesen ismeretlen és csak a 3.7 fejezetben fogjuk tárgyalni.

A 31-es regiszter a sprite–háttérjelütközéseket rögzíti úgy, hogy a sprite-hoz tartozó bit bekapcsolódik. A 2-es sprite ütközésekor tartalma: %0000 0010 lesz.

Lekérdezés után ezt is törölni kell, mert különben téves információhoz vezethet.

Ebben az esetben egyidejűleg a 25-ös regiszter 1. bitje kapcsolódik be, hogy szükség esetén megszakítást (IRQ) válthasson ki.

3.6 Szöveg/jelkészlet

3.6.1 Képernyőoldal szerkesztése szöveg üzemmódban

Mint már említettük, a számítógépnek az adott képernyői képet minden 1/20 másodpercben újra és újra elő kell állítania (ld. 3.7 fejezet). Ahhoz, hogy ezt megtehesse, a képernyőn levő jelekre vonatkozóan információkat tárol a képernyőtárban. Ehhez mintegy 1 k tárkapacitással rendelkezik, de ebből a képernyőtárolás valójában csak 40×25 = 1000 byte-ot igényel. Alapállapotban a $0400–$07FF (1024–2047) tártartományban helyezkedik el. Eltolási lehetőségeit a 3.3.2 fejezetben ismertettük. Nagyfelbontású és többszínű grafikus üzemmódban színtárként használjuk.

Minden egyes képernyői jelnek kódja van, ami a képernyői kivitelkor a képernyőtár megfelelő helyére kerül. A kódolás lényege az ASCII-kódok révén bizonyára már ismerős. A képernyőkódokat azonban más rendszer szerint képezzük. A CBM-kézikönyv Függelékében található ASCII-táblázatban

látható, hogy néha ugyanannak a jelnek különböző értékei vannak és egyidejűleg ún. vezérlő értékkel is rendelkeznek, amit azonban nem a képernyői kivitelkor használunk.

A képernyőkódok ezzel szemben egyértelműek és hézag nélkül vannak elosztva, továbbá megkülönböztetjük a jel normál és inverz alakját úgy, hogy az inverz jel kódja = a normál jel kódja+128.

Mint tudjuk, az átkapcsolás billentyűzetről az RVS/ON (=CHR$ (18)) és az RVS/OFF (=CHR$ (146)) billentyűkkel történik. A Függelékben található egy képernyőkód-táblázat, ami a normál kódokat tartalmazza. Az előzőek értelmében ezekhez 128-at adva rendre megkapjuk a jelek inverz kódját.

3.6.1.1 Egyszínű szöveg üzemmód

A képernyőkódok mellett minden jelhez még egy jelszínt is tárolni kell, mivel elméletileg azok egy másik színt is felvehetnek. Erre szolgál a színtár, ami ugyanakkora, mint a képernyőtár és a jelekre vonatkozó színinformációkat tárolja. A $D800–$DFFF (55296–56295) tártartományban fekszik és többszínű grafikus üzemmódban is használjuk. Minden byte-ja a képernyőtárban hozzá tartozó jel színét határozza meg. A képernyőtár és a színtár felépítése megegyezik.

Ezzel szemben a képernyő háttérszínét a VIC egyetlen regisztere, a 33-as vezérli. Tárcíme 53248+33 és például a

```
POKE 53248+33, 0 : REM HÁTTÉR FEKETE
```

utasítással módosítható (0 = a fekete szín kódja).

3.6.1.2 Többszínű szöveg üzemmód

A jelábrázolás ezen módjáról a 3.2 fejezetben már beszéltünk. Ismételjük át újból ezt a részt!

3.6.1.3 Bővített háttérszín (extended colour) üzemmód

Ebben az üzemmódban lehetőségünk van arra, hogy minden képernyőn megjelenő jelhez – négy színből – külön háttérszínt rendeljünk. Ehhez négy regiszter áll rendelkezésünkre:

Háttérszín száma	Regiszter
0	VIC 33
1	VIC 34
2	VIC 35
3	VIC 36

Tartalmuk POKE-utasítással módosítható. A 0. háttérszín az általános háttérszínregiszterből származik.

Az üzemmód bekapcsolása a következő utasítássorral történik:

```
POKE 53248+17, PEEK (53248+17) OR 4*16
```

Ezzel bekapcsoltuk a 17-es VIC-regiszter 6-os bitjét. Kikapcsolni a

```
POKE 53248+17, PEEK (53248+17) AND 256–4*16
```

utasítássorral lehet.

Az egyes jelek háttérszínének meghatározásához a következőket kell tudnunk: A video-RAM byte-jainak, amelyek eredetileg a képernyőkódokat tárolják, felső két bitje határozza meg a jelhez tartozó háttérszín származási regiszterét. Így a képernyőkódok ábrázolására már csak 6 bitünk marad, ami max. 64 különböző értéket vehet fel. Ebben az üzemmódban ezért csak 64 különböző jel áll rendelkezésünkre. Áldozatul esnek a grafikus jelek, kisbetűs üzemmódban a nagybetűk és az inverz jelek.

Ezek kódjait felhasználva is csak a megengedett 64 jel valamelyike jelenik meg, változó háttérszínnel. Az érintett képernyőkódokat legjobban a Függelékben levő táblázat szemlélteti. A következő hozzárendelések érvényesek.

Felső bitek (6/7)	Képernyőkódok	Háttérszín száma
00	000–063/$00–$3F	0
01	064–127/$40–$7F	1
10	128–191/$80–$BF	2
11	192–255/$C0–$FF	3

Hasonlítsuk össze a Függelék táblázatával.

Egy példa: a képernyő bal felső sarkában egy B betűt szeretnénk megjeleníteni, sárga (=7) háttérszínnel. Az 1-es háttérszínregisztert lefoglaljuk

a sárga szín számára: POKE 53248+34, 7, majd a képernyőtár első bytejába beírjuk a B betű kódját, hozzávéve a felső bitek értékét, ami most 64 (az 1-es háttérszín miatt, táblázat 2-es sor). Ehelyett PRINT-utasítást is alkalmazhatunk (SHIFT/B=CHR$(98)). Programban a két változat a következőképpen néz ki:

```
1 REM *** P 3 ***
2 REM
3 PRINT"♥"
10 V=53248:REM A VIC KEZDOCIME
20 POKE V+17,PEEK(V+17) OR 4*16:REM BHSZ UZEMMOD BE
30 POKE V+34,7:REM 1.HATTERSZIN=SARGA
40 POKE 1024,2+64:REM JEL A BAL FELSO SAROKBA
```

vagy

```
1 REM *** P 4 ***
2 REM
10 V=53248:REM A VIC KEZDOCIME
20 POKE V+17,PEEK(V+17) OR 4*16:REM BHSZ UZEMMOD BE
30 POKE V+34,+7:REM 1.HATTERSZIN=SARGA
40 PRINT CHR$(98) REM JEL AZ AKTUALIS KRZ.POZICION
```

A programok futása után a bővített háttérszín üzemmódban maradunk, tovább kísérletezhetünk. Az eredeti állapot visszaállításának módját már ismerjük.

3.6.2 A jelkészlet

A szokatlan grafikai változatokon kívül a CBM 64 még további értékes lehetőségekkel rendelkezik. Az egyik a jelkészlet szoftver útján való módosítása. Ennek nagy jelentősége van a játékprogramokban. A sprite-ok mellett nagyban hozzájárul ahhoz, hogy a 64-esen futtatott játékprogramok grafikusan kidolgozottak, látványosak és mégis gyorsak lehetnek. Enélkül egy komoly játékprogram el sem képzelhető.

Elsőként a fogalom jelentése: Jelkészlet alatt értjük azon jelek összességét, amelyeket szöveg üzemmódban bármely billentyű vagy billentyűkombináció lenyomásával a képernyőn elő tudunk állítani. Ezek betűk, számok, különleges és grafikus jelek.

A különböző jelek alakját természetesen ismernie kell a számítógépnek, ezért azokat valahol, valamilyen formában tárolnia kell. Ezenkívül rendszerezni kell oly módon, hogy pl. a B billentyű lenyomása a képernyőn is B betűt eredményezzen.

A CBM 64 ebben illetékes tárterületét úgy alakították ki, hogy programból hozzáférhető, vagyis tartalma kiolvasható és pl. máshová átmásolható legyen. Erről részletesen olvashattunk a 3.3.1 fejezetben.

Számítógépünket utasíthatjuk arra, hogy ezentúl a jelek alakját pl. a $2000 (=8192) tárcímtől kezdődő RAM-ból olvassa ki, aminek tartalma módosítható.

Ha a ROM tartalmát átmásoljuk a RAM-ba, először semmi változást nem észlelünk, hiszen a tárolt információk nem változtak, csak áthelyeződtek. Az egyes byte-okat módosítva azonban a képernyőn megjelenő jelek alakja is megváltozik.

Ahhoz, hogy a jelek alakját módosíthassuk, ismernünk kell tárolásuk módját.

Számítógépünk négyféle jelkészlettel rendelkezik, egyenként 128 jellel. Ezekből egyszerre kettőt alkalmazhatunk a képernyőn. Elnevezésük a következő:

I/1	– normál	– nagybetű/grafikus jelek
I/2	– inverz	– nagybetű/grafikus jelek
II/1	– normál	– kisbetű/nagybetű
II/2	– inverz	– kisbetű/nagybetű

Egyidejűleg az I/1–I/2 és II/1–II/2 használható. Az I/II átváltás billentyűzetről a SHIFT/C= billentyűkombinációval, programból a 14-es és 142-es ASCII-kódokkal történik (142 = 14+128):

PRINT CHR$ (14) = II. jelkészlet

PRINT CHR$ (142) = I. jelkészlet

Az átkapcsolás a

PRINT CHR$ (8)

utasítással megakadályozható, ami a

PRINT CHR$ (9)

utasítással oldható fel.

Az 1/2 átkapcsolás az RVS-ON/RVS-OFF billentyűkkel történik.

A jelkészlettárban természetesen mind a négy jelkészletet fel kell sorolni. Összesen tehát 4*128 = 512 jel módosítására van lehetőségünk. (A képernyőn ebből egyszerre csak 256 ábrázolható!)

Azt is tudjuk már, hogy a jelek a képernyőn egy 8×8-as pontmezőben helyezkednek el. A tárban ezt a 8×8 = 64 pontot kell megfelelően rögzítenünk. Ez a nagyfelbontású grafikához hasonlóan valósul meg. A jel minden pontjához a tár 1 bitje tartozik. Így minden jelhez egy 8 byte-ból álló ún. bitminta készíthető, aminek minden sora 1 byte-ot jelent a tárban. Ennek megfelelően egy bekapcsolt bit a tárban 1 bekapcsolt pontot jelent a képernyőn. Tárbeli elhelyezkedését a következőképpen kell elképzelnünk:

Bit	7	6	5	4	3	2	1	0
0. byte	.	.	.	.	.	.	.	.
1. byte	.	.	.	.	.	.	.	.
2. byte	.	.	.	.	.	.	.	.
3. byte	.	.	.	.	.	.	.	.
4. byte	.	.	.	.	.	.	.	.
5. byte	.	.	.	.	.	.	.	.
6. byte	.	.	.	.	.	.	.	.
7. byte	.	.	.	.	.	.	.	.

A jelkészlettár 512 ilyen jellegű, egymás után következő bitmintából áll, tehát összesen 4 k (=4096 byte) tárkapacitást köt le. Alapállapotban a $D000–$DFFF (53248–57344) ROM-tartományban helyezkedik el. Ez a terület BASIC-ból nem olvasható.

Szemléltetésül bemutatjuk a normál B betű bitmintáját:

Bit:	7	6	5	4	3	2	1	0
0. byte :	.	*	*	*	*	.	.	.
1. byte :	.	*	*	.	.	*	*	.
2. byte :	.	*	*	.	.	*	*	.
3. byte :	.	*	*	*	*	.	.	.
4. byte :	.	*	*	.	.	*	*	.
5. byte:	.	*	*	.	.	*	*	.
6. byte:	.	*	*	*	*	.	.	.
7. byte :	.	.	.	.	.	.	.	.

A 8 byte tartalma a következő:

0. byte = 0111 1000 = $78 = 120
1. byte = 0110 0110 = $66 = 102
2. byte = 0110 0110 = $66 = 102
3. byte = 0111 1000 = $78 = 120
4. byte = 0110 0110 = $66 = 102
5. byte = 0110 0110 = $66 = 102
6. byte = 0111 1000 = $78 = 120
7. byte = 0000 0000 = $00 = 000

Ezt a nyolc értéket tartalmazza a normál B betű számára fenntartott tárhely.

Többszínű üzemmódban (szokás szerint) minden két bit jelent a képernyőn egy dupla széles pontot. Ezáltal a felbontás 4×8 pontra csökken. Lényege ugyanaz, mint amit már a grafikánál és a sprite-oknál is elmondtunk, ezért itt csak egy kis ábrán szemléltetjük a pontok és bitek összefüggését:

Bit:	7 6 5 4 3 2 1 0
0. byte :	<-> <-> <-> <->
1. byte :	<-> <-> <-> <->
2. byte :	<-> <-> <-> <->
3. byte :	<-> <-> <-> <->
4. byte :	<-> <-> <-> <->
5. byte :	<-> <-> <-> <->
6. byte :	<-> <-> <-> <->
7. byte :	<-> <-> <-> <->

A következő probléma a bitminták táron belüli helyzetének meghatározása. A számítás alapja a képernyőkód, amit minden képernyőn levő jelre vonatkozóan a képernyőtár rögzít.

A többi viszonylag egyszerű: mivel minden bitminta 8 byte-ból áll, a tárcíme úgy kapjuk meg, hogy a képernyőkódot megszorozzuk 8-cal és hozzáadjuk a jelkészlettár kezdőcímét. A kezdőcím megállapításakor ügyelnünk kell arra hogy az I-es és II-es jelkészletet megkülönböztessük. A kisbetűs (II) jelkészle b-je 2 k távolságra van a nagybetűs (I) jelkészlet B betűjétől és mindkett képernyőkódja 2. Ez utóbbi nem tévesztendő össze a kisbetűs jelkészlet B betűjével, aminek a képernyőkódja 65. Az eredeti jelkészlet kezdőcímei:

– nagybetű/grafikus jelkészlet: $D000 (53248)
– kisbetű/nagybetű: $D800 (55296)

A fentiekből következik a bitminta címét adó képlet:

cím = báziscím+8*képernyőkód.

A normál B betűre:

cím = $D000+8*2 = $D010 = 53256.

A 4. fejezetben részletesen megismerkedünk a tanultak alkalmazásával is.

3.7 IRQ-lehetőségek

Számítógépünk egyik rendkívül értékes tulajdonsága a sokféleképpen alkalmazható megszakítás (interrupt). Ezalatt a program tetszőleges helyen kijelölt, vezérelt megszakítását értjük, amit valamilyen előre meghatározott esemény idéz elő. Megszakítás kiváltásakor a belső processzorprogram a tároló egy indirekt módon címezhető helyére ugrik és végrehajtja az ott található megszakító alprogramot. Ezután visszaugrik a főprogramba (az elágazás helyére) és folytatja annak feldolgozását – a következő megszakításig.

Ne ijedjünk meg. Akkor is nyugodtan tovább olvashatunk, ha eddig még sosem hallottunk a megszakításról, vagy nem ismerjük megfelelően a gépi nyelvű programozást. Minden itt ismertetésre kerülő, megszakításon alapuló dolog ezek nélkül a – csak assemblerben programozható – trükkök nélkül is működik. A teljesség kedvéért akkor is érdemes ezeket a részeket átolvasni, ha egyébként ezen a területen nem vagyunk járatosak. A végén a BASIC-ben adódó lehetőségekre is kitérünk.

Az IRQ tehát egy szándékos és programtechnikailag tervezett megszakítás, ami nem jelenti a program összeomlását vagy végleges megszakadását. IRQ elvileg BASIC-ben is megvalósítható (ld. SUPERGRAPHIK 64), de amikről itt beszélünk, az a CPU hardverszinten beszabályozott, szoftverrel vezérelt megszakításai. A mi készülékünk mikroprocesszora négy különböző megszakításra képes:

– Reset.

- NMI (Non Maskable Interrupt).
- BRK (break).
- IRQ (Interrupt ReQuest).

a) *Reset*

Szoftver útján nem iktatható ki. Bekapcsolás után kerül kiváltásra és feladata a gép alapállapotának beállítása. A folyamat végén megjelenik a képernyőn az ismerős felirat:

```
**** COMMODORE 64 BASIC V2 ****
64 K RAM SYSTEM 38911 BASIC BYTES FREE
READY
```

b) *NMI*

Nem maszkolható, azaz szintén feltétlen megszakítást jelent a RESTORE-billentyű lenyomásakor. RS 232 interface alkalmazásakor is szükséges. Indirekt ugrási címét a $318/$319 (792/798) rekeszek tárolják, ami szükség esetén módosítható.

c) *BRK*

Ez egy ún. szoftvermegszakítás, ami assemblerprogramból működtethető. Akkor kerül kiváltásra, ha processzor a $00 BRK-kódra bukkan. Indirekt ugrási címét a $316/$317 (790/791) rekeszek tárolják.

d) *IRQ*

Végre a lényeg! Az előzőekkel nem kell tovább foglalkoznunk, csak a teljesség kedvéért említettük meg őket. Számunkra az egyetlen érdekes megszakítási lehetőség az ún. maszkolható megszakítás. Maszkolni annyit jelent, mint a megszakítás okát előírni, azaz szoftver útján meghatározhatjuk, hogy kiváltson-e megszakítást valamilyen esemény, vagy ne. (Elnevezése is erre utal. Nyers fordításban: IRQ=Interrupt ReQuest=megszakítás kérése.)

Az IRQ-t ebből a szempontból célszerű az egyéb megszakításoktól megkülönböztetni, ezért amikor erről beszélünk, az "IRQ" egyébként is elterjedt kifejezést fogjuk használni (a magyar változat készítői).

A fentiekből következik, hogy az IRQ tetszés szerint ki is kapcsolható. Erre két assemblerutasítás létezik:

- SEI (SEt Interruptflag = a megszakításjelző bekapcsolása)
- az IRQ megakadályozása,
- CLI (CLear Interruptflag = a megszakításjelző törlése)
- az IRQ engedélyezése.

(Jelzőn (flag) egy meghatározott regiszter meghatározott bitjét értjük.)

A Commodore 64-es speciális regisztereiben egyidejűleg kiválaszthatjuk, hogy mely események váltsanak ki IRQ-t és melyek ne. Így lehet pl. a CIA időzítőjével megszakítást előidézni, amit a BASIC-operációsrendszer használ úgy, hogy minden 1/60 másodpercben megszakad a normál programfutás és a ROM-IRQ-rutinba ugrik. (Indirekt címét a $314/$315 (788/789) rekeszek tárolják.) Itt van a kurzor villogtatása, a belső óra (TI$) beállítása és a billentyűzet (pl. RUN-STOP) lekérdezése.

Egyéb eseményekkel is válthatunk ki IRQ-t. Ezek közül számunkra a következők az érdekesek:

- rasztersor-felépítés,
- fényceruza-impulzus,
- sprite–sprite-ütközés,
- sprite–háttérjel-ütközés.

Ezeket a lehetőségeket úgy tudjuk kihasználni, ha a $314/$315 (788/798) rekeszekben található címet (mutató) egy saját megszakító alprogram kezdőcímére módosítjuk. Ennek programtechnikai megoldását a 4. fejezetben ismertetjük.

Az IRQ előnye a felhasználó számára az, hogy az eseményt azonnal kijelzi anélkül, hogy várnia kellene a következő lekérdezésig (az IRQ mellett az esemény bekövetkezése a programfutás alatt természetesen egy egyszerű lekérdezéssel is ellenőrizhető). Ennek a rasztersor-IRQ-nál van különösen nagy jelentősége, mert a folyamatoknak ebben az esetben nagyon gyorsan kell lezajlaniuk.

Nos, ezek után térjünk rá a részletekre!

3.7.1 Képernyő-regisztersorok

Számítógépünk legkevésbé érthető, de éppen emiatt legvonzóbb tulajdonsága a rasztersor-IRQ. Számos csodálatos hatást érhetünk el vele: több háttérszín, 8-nál több sprite, a grafikus- és szöveg-képernyő kombinálása... stb. Mielőtt azonban erre a 4. fejezetben rátérnénk, tisztáznunk kell néhány dolgot. Először is azt, hogy mit értünk rasztersor alatt.

Mindenekelőtt tudnunk kell, hogy jön létre a kép a tv vagy monitor képernyőjén. A képernyő tulajdonképpen a képcső (ami nem más, mint egy elektronsugárcső) homlokfelülete, amit belülről foszforbevonattal láttak el. Emögött helyezkedik el az ún. lyukmaszk (vagy lyukraszterlemez), ami egy lyukacsos fémlemez.

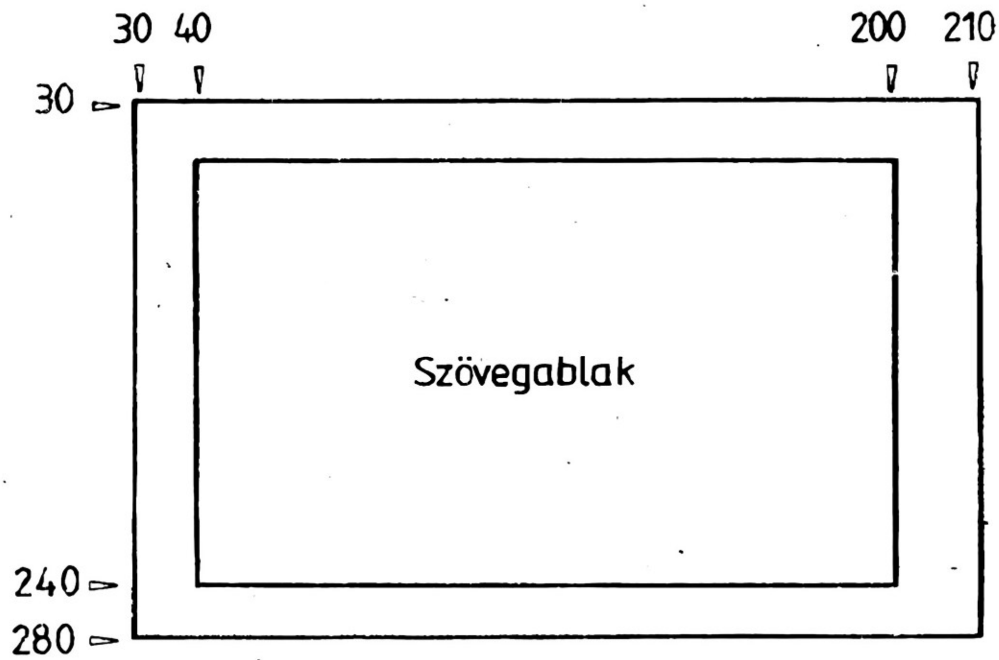
Ha az elektronsugár egy elektronja nekiütődik a foszforrétegnek, a képernyőn ezen a helyen egy világító pont fog megjelenni. Az elektronsugár sorról sorra átmegy a lyukmaszk minden pontján és a vezérléstől függően bekapcsolja a hozzá tartozó pontot. Ez egy rendkívül gyors folyamat. Az elektronsugár másodpercenként 20-szor állít elő új képet, vagyis hússzor megy át a lyukraszter minden pontján. Az ilyen gyorsan egymás után felvillanó képek a folyamatosság benyomását keltik.

Az elektronsugarat a képcső végén található bonyolult berendezések vezérlik. Az információk azonban, amelyek meghatározzák, hogy egy pont milyen erősséggel villanjon fel, más forrásból származnak.

Normál tv esetében az adóállomások az éteren keresztül sugározzák a megfelelő jeleket. A mi esetünkben ezeket a számítógépnek kell előállítania, tehát sorról sorra és pontról pontra haladva "ki/be"-jeleket küldenie. Ezt a feladatot a CBM 64 esetében a VIC látja el.

Általában az egész folyamat belsőleg szabályozott anélkül, hogy a programozó befolyásolhatná, vagy egyáltalán részt vehetne benne.

A 64-esnél azonban ez másképpen van: Lehetőségünk van ugyanis szoftver útján megállapítani azt, hogy a VIC éppen melyik rasztersort állítja elő. Ez a 18-as VIC-regiszterből olvasható ki. Mielőtt azonban erre rátérnénk, még valamit el kell mondanunk: Azt mindenki tudja, hogy a számítógép által használt képernyőablak kisebb, mint a teljes képernyő. Ebből egyértelműen adódik, hogy a kép felépítésekor a rasztersor keretbe eső részét is fel kell építeni. Ráadásul az elhajlás miatt az elektronsugár túlmegy a képernyő szélén. Emiatt a raszterkoordináták eltérnek a megszokott grafikus- vagy sprite-koordinátáktól. Ennek szemléltetésére nézzük a következő ábrát.



Az elektronsugár a legfelső sorban indul, de 0. vagy 1. sor nem létezik, hanem a képernyő felső széle már a 30. ($1E) sorban van. (Ez az érték tv-ről tv-re könnyen változhat.) Az alsó képernyőszél kb. a 280. ($118) sorban van. A tulajdonképpeni képernyőablak a 40. sortól a 240. sorig terjed.

Mint látjuk, a VIC a képernyőablakot, a pontfelbontásnak megfelelően, pontosan 200 sorra osztja. A botkormánnyal (joystick) foglalkozó fejezetünkben majd látni fogjuk, hogy az oszlopfelbontás esetében ez nem ilyen egyszerű. Mint már említettük, aktívan részt vehetünk az eseményekben és tájékozódhatunk a mindenkori állapotokról. Mit jelent ez a gyakorlatban?

Először is kiolvashatjuk a VIC 18-as regiszteréből annak a rasztersornak a számát, ahová éppen jelet küld. Mivel a regiszter a 8 bitjével csak 256 különböző értéket vehet fel és legalább 280-ra van szükség, a 17. regiszter 7. bitjét is igénybe vesszük (ld. 3.1 fejezet). Ezek együtt adják az aktuális rasztersor számát, amit egyszerű kiolvasással mi is megtudhatunk.

Mivel másodpercenként 20-szor kell új képet előállítani és képenként legalább 280 sort kell felépíteni, könnyen kiszámíthatjuk, hogy egy sor kb. 1/(20*280) = 0,00018 s = 180 μs alatt jön létre (1 μs a másodperc egymilliomod része), vagyis 1 másodperc alatt 5600 sor épül fel. Ha belegondolunk, hogy egy sor milyen nagyszámú pontból áll, érzékelhetjük, mennyi idő jut 1 pontra (kb. 0,89 μs). Emiatt szinte lehetetlennek tűnik ezeknek a folyamatoknak a szoftverszintű kezelése, hiszen még assemblerben is 1/500 000 (=2 μs) másodpercet vesz igénybe egy utasítás feldolgozása. Néhány utasítás időszükséglete pedig ennek még 3,5-szerese is lehet. Ilyen körülmények között a sorok vezérléséhez szinte állandóan le kellene kérdeznünk az elektronsugár helyzetét.

Egy rendkívül érdekes megoldás azonban mindezt szükségtelenné teszi. A VIC-et ugyanis beprogramozhatjuk arra, hogy ha a képfelépítésben egy adott sorba ér, IRQ-t váltson ki. (Adott pontra vonatkozóan, éppen a nagyon kicsiny pontidő miatt, ezt nem írhatjuk elő.)

Ebből a célból az általunk kiválasztott sor számát beírhatjuk ugyanabba a 18-as regiszterbe, ahonnan egyébként a sugár aktuális pozícióját állapítjuk meg. A 17-es regiszter 7. bitje felső bit most is. Közölnünk kell továbbá a számítógéppel (ill. a VIC-kel), hogy amikor a kijelölt sort elérte, azonnal hajtsa végre a megszakítást. Ez a 25-ös és 26-os regiszterek igénybevételével érhető el.

Az első az IRR (Interrupt Request Registern = megszakítást kérő regiszter), aminek az a feladata, hogy rögzítse, ha az esemény bekövetkezett. (Esetünkben, ha az elektronsugár elérte a kijelölt rasztersort.)

A következő hozzárendelések érvényesek:

0. bit =	1 – az aktuális rasztersor száma = a 18. regiszterbe beírt értékkel,
1. bit =	1 – sprite–háttérjel-ütközés,
2. bit =	1 – sprite–sprite-ütközés,
3. bit =	1 – fényceruza-impulzus,
4–6. bitek =	nem használt,
7. bit =	1 – ha az alsó négy bit valamelyike 1.

A programozó tehát a 7. bit lekérdezésével megállapíthatja, hogy a 0–3. bitek közül egyenlő-e valamelyik 1-gyel, vagyis hogy a megszakításkérést e négy ok valamelyike váltotta-e ki. Ez azért érdekes, mert, mint mondtuk, azt más események is előidézhetik.

Ezt a regisztert minden lekérdezés után vissza kell állítani, különben az interrupt-rutin feldolgozása után rögtön új megszakítás következik be, és ezzel a normális programfutás lehetetlenné válik. A program összeomlik. A visszaállítás úgy lehetséges, hogy a regiszterbe visszaírjuk azt az értéket, amit eredetileg tartalmazott. Ebből azonban a VIC még mindig nem tudja, miáltal kell megszakítást kiváltania. Ez a feladata a 26-os, ún. IMR- (Interrupt Masc Register) regiszternek. A bitek foglaltsága a 7. bit kivételével ugyanaz, mint a 25-ös regiszteré.

Ennek minden egyes bekapcsolt bitje azt jelenti, hogy az általa kijelölt eseménymegszakítást kiváltó ok lesz.

Egy példa: Azt akarjuk, hogy amikor a VIC eléri a 100. rasztersort, szakítsa meg a programot és a processzor hajtsa végre a megszakítórutint. Első dolgunk az, hogy a 18. regiszterbe beírjuk a 100-at (a 17. regiszter 7-es bitje = 0!), töröljük a 25-ös regisztert azáltal, hogy tartalmát kiolvassuk és ismét visszaírjuk, majd bekapcsoljuk a 26-os regiszter 0. bitjét. A folyamat úgy fog lezajlani, hogy amikor a képernyő felépítése a 100. rasztersorba ér, a 25. regiszter 0. bitje bekapcsolódik, ami megszakításkérést jelent. Miután a 26-os regiszter 0. bitje is 1, a megszakítás engedélyezett, tehát azt a VIC végrehajtja. Ha a 26-os regiszterbe több megszakítást kiváltó okot is engedélyezünk, akkor a 25-ös regiszter tartalmának kiolvasásával dönthetjük el, hogy éppen melyik következett be.

3.7.2 A fényceruza

Mielőtt erre a fejezetre rátérnénk, ismételjük át, amit a 3.7.1 fejezetben a képelőállításról tanultunk, mert ezek pontos ismerete nélkül a most következő dolgokat nem fogjuk megérteni.

A Commodore 64-es külső vezérlőegységek csatlakoztatását is lehetővé teszi. Erre a célra a gép jobb oldalán két csatlakozást, ún. vezérlőportot (controlport 1, controlport 2) találunk. Ezek a különböző perifériák (borkormány, potméterek, fényceruza, vagy egyéb, esetleg saját készítésű mérőeszközök, pl. nedvességmérő, impulzuskeltő, termométer) csatlakozási helyei.

A csatlakozók foglaltságát nem kell feltétlenül ismerni ahhoz, hogy – mondjuk – a botkormányt megfelelően csatlakoztatni és működtetni tudjuk. Az azonban fontos, hogy a fényceruza bemenete megegyezzen az 1-es portra csatlakoztatott botkormány tűzgombbemenetével. A tűzgomb alkalmas arra, hogy az 1-es porton keresztül megszakítást váltson ki, ami nem csak játéknál használható nagyszerűen.

Nézzük magát a fényceruzát:

Fényceruza alatt egy olyan (ceruzához hasonló) kis eszközt értünk, ami alkalmas arra, hogy a tv képernyője és a felhasználó között közvetlen kapcsolatot teremtsen. Képes a képernyő egy pontjának helyzetét a számítógép számára egyértelműen meghatározni. Egyszerűbben fogalmazva: a fényceruzát a képernyő kiválasztott pontjára helyezve, a számítógép meg tudja állapítani a pont helyzetét. Működése nagyon egyszerű. Az 1-es vezérlőportra csatlakoztatott fényceruzával rámutatunk a képernyő valamelyik pontjára. Teljesen mindegy,

hogy ez a pont a képernyőablakon belül, vagy azon kívül van-e. A számítógép képes arra, hogy ezután azonosítsa a pont helyzetét, vagyis meghatározza annak koordinátáit. Ha ezt a programunkban lekérdezzük, megállapíthatjuk, hogy azon a bizonyos helyen van-e pl. betű vagy grafika. Mindez fordítva is lehetséges, tehát a fényceruzával rajzolhatunk is a képernyőre. Ezenkívül kényelmes kurzorvezérlő eszközként is használhatjuk... stb.

Ezek után térjünk rá a dolog technikai oldalára.

Hogyan állapítja meg a számítógép a fényceruza helyzetét? Mint már tudjuk, a képernyő apró pontok sokaságából áll, amelyek a lyukmaszkon áthaladó elektronsugár hatására minden 1/20 másodpercben felvillannak.

A fényceruza ezt a felvillanást érzékeli. (Éppen ezért, ha a fényceruzával akarunk dolgozni, ne válasszunk fekete háttérszínt és a fényerőt se állítsuk túl alacsonyra.) Amikor tehát az elektronsugár a kijelölt pontra ér, a fényceruza érzékeli a pont felvillanását és jelzést küld a számítógépnek.

A számítógép az elektronsugár útját folyamatosan követi, ami azt jelenti, hogy mindig tudja azt, hogy éppen melyik sorban és melyik oszlopban van. Ezért amikor a fényceruzától jelzést kap, azonnal rendelkezésére állnak ezek az adatok. Amikor a jel eléri a VIC-et, az aktuális rasztersor és raszteroszlop (eddig ezeket a fogalmakat még nem ismerjük) száma, mint X és Y koordináták, bekerülnek a 19-es és 20-as VIC-regiszterekbe. Innen a program kiolvashatja és feldolgozhatja, pl. úgy, hogy egy pontot rajzol a kijelölt helyre.

Itt néhány szó erejéig ki kell térnünk a koordináta-rendszerre.

A rasztersorok szerinti (Y irányú) beosztást a 3.7.1 fejezetből már ismerjük. Az X irányú, vagyis a raszteroszlopok szerinti beosztás valamivel bonyolultabb. A működtethető raszterpontok száma feleannyi, mint a grafikus pontok száma. Egy raszterpont tehát két grafikus pontot ad. Szélső értékei a következők: a képernyő bal széle kb. 30, a jobb széle pedig kb. 210. A szabályos képernyőablak kb. 40–200-ig terjed, összesen 160 = 320/2 raszterponttal.

Tételezzük fel, hogy fel akarjuk használni a 19-es és 20-as VIC-regiszterek tartalmát. A kijelölt helyre egy pontot szeretnénk rajzolni. Ahhoz, hogy ezt megtehessük, először át kell számítanunk a raszterkoordinátákat grafikus koordinátákká. A következő képleteket használjuk:

$$X_g = (X_r - 40) * 2$$

$$Y_g = (Y_r - 40)$$

ahol X_g és Y_g a grafikus koordinátákat, X_r és Y_r a raszterkoordinátákat jelentik. Az így kiszámított értékekkel most már a szokásos módon dolgozhatunk tovább.

Ez egy egyszerű, BASIC-ben is megvalósítható eljárás és számítógépünk itt is lehetőséget nyújt a megszakítási technika alkalmazására. Amikor a fényceruza jelet küld, a 25-ös VIC-regiszter 3. bitje bekapcsolódik (1-esre vált). Amennyiben egyidejűleg a 26-os regiszter 3. bitje is 1, a megszakítás engedélyezett. A főprogram végrehajtása megszakad és a program a megszakítási rutin feldolgozásával folytatódik.

A 4. fejezetben erre is találunk példaprogramot.

3.7.3 A sprite-ok ütközése

Ha elolvastuk a 3.5.4.4 fejezetet, tudjuk, hogy a sprite-ok egymással és a háttérjelekkel való ütközését a 30/31-es VIC-regiszterek jegyzik. Ezekben minden bit egy sprite-ot jelent. Egy sprite ütközésekor a hozzá tartozó bit bekapcsolódik.

Az ütközések jelzésére még egy lehetőségünk van, ami ugyancsak a megszakítási technikán alapul. A 25-ös és 26-os regiszterek bevonásával a sprite-ütközések esetére megszakítást írhatunk elő. Ebben az esetben külön kell kezelnünk a sprite–sprite- és a sprite–háttérjel-ütközéseket, ezért erre a célra mindkét regiszterben két bitünk van. Az 1. bit a sprite–háttérjel-, a 2. bit pedig a sprite–sprite-ütközéseket jelzi. Az előző fejezetből már tudjuk, hogy a megszakítás csak akkor következik be, ha megfelelő bitek a 25-ös és 26-os regiszterben is egyidejűleg bekapcsolt állapotban vannak. Ne felejtkezzünk el az IRQ-címek módosításáról és a 25-ös regiszter törléséről.

4. FEJEZET

A grafika programozásának alapjai

Nos, eleget foglalkoztunk számítógépünk rejtelmes grafikai képességeivel ahhoz, hogy végre rátérhessünk a gyakorlati megvalósításra. Az egész tudományunk semmit sem ér addig, míg a grafikát vagy a sprite-okat be sem tudjuk kapcsolni. Ezt a fejezetet úgy építettük fel, hogy egyes részei a 3. fejezet egy-egy témájához kapcsolódnak, alkalmazási és programozási példával kiegészítve az ott leírtakat. A mintaprogramokat igyekeztünk BASIC-ben írni, de ahol mindenképpen gépi kódú programozásra volt szükség, ott a gépi kódú program BASIC-betöltőjének listáját is közöljük. A programok sok magyarázó szöveget (REM-es sorok) tartalmaznak. Ezeket a gyorsabb programfutás és a kisebb tárigény érdekében a bevitelkor el lehet hagyni. A fontos és új dolgokat tartalmazó programrészekhez külön magyarázatot fűzünk.

A különböző rutinok BASIC-nyelvű megfelelőjét csak segítségképpen közöljük, mert ezek gépi kódban összehasonlíthatatlanul gyorsabbak és sokkal jobban megfelelnek a céljainknak. Említettük már, hogy érdemes elsajátítani a gépi kódú programozást. Aki a BASIC-et többé-kevébé már ismeri, annak ez sem fog komoly nehézséget okozni.

4.1 Szöveg és grafika a kisfelbontású képernyőn

A grafikai munka legegyszerűbb esete, amikor a képet szöveg üzemmódban, a billentyűk homloklapján található grafikus jelekből állítjuk össze. Ehhez csak egy kis képzelőerő és a megfelelő billentyűk kellenek. Ezzel az egyszerű eljárással is nagyon szép grafikákat készíthetünk.

A grafikus jelek előállítása:

a) *Vezérlés a billentyűzetről*

A billentyűk homloklapján két-két grafikus jelet találunk. Ezek közül a jobb oldalit a SHIFT-, a bal oldalit a C=-billentyű egyidejű lenyomásával választjuk ki. Ezenkívül a CTRL/RVS ON billentyűkombinációval a jeleket inverz alakban is ábrázolhatjuk. Ezzel egyenértékű a

```
PRINT CHR$ (18)
```

utasítás, amit a

```
PRINT CHR$ (146)
```

utasítással, vagy a CTR/RVS OFF billentyűkombinációval hatástalaníthatunk.

Színek:

A 16 különböző jelszín szintén kiválasztható a billentyűzetről (ld. Függelék). Az első 8 színt a CTRL-billentyű, a 8–15 színeket pedig a C=-billentyű és az 1...8 billentyűk egyidejű lenyomásával állíthatjuk elő.

b) *Vezérlés PRINT-utasítással*

Mint tudjuk, minden jelnek van egy ún. ASCII-kódja. Ebben az esetben a normál és inverz alak között nem teszünk különbséget. Az egyes jelek ASCII-kódját az ASC-utasítással kérdezhetjük le, a következő módon (pl. az A betűre):

```
PRINT ASC("A")
```

vagy

```
Z$="A":PRINT ASC (Z$)
```

RETURN után a képernyőn megjelenik az A betű ASCII-kódja, a 65.

A lekérdezés fordítva is lehetséges. Ha nem tudjuk, hogy egy ASCII-kódhoz melyik jel tartozik, a

```
PRINT CHR$ (65)
```

vagy a

```
C=65:PRINT CHR$ (C)
```

utasítással kiírathatjuk a képernyőre.

A kódolás előnye pl. az, hogy a képernyőn megjelenő jeleket nem csak közvetlenül, hanem valamilyen számítási eljárással is meghatározhatjuk. Ezenkívül a jelek nagyon könnyen összehasonlíthatók.

Színek:

A színeknek is van ASCII-kódjuk. Mivel CBM 64-es kézikönyvében csak az első 8 színét sorolták fel, a következőkben közöljük a teljes listát:

ASCII	Szín
144	fekete
5	fehér
28	piros
159	türkiz
156	ibolya
30	zöld
31	kék
158	sárga
129	narancs
149	barna
150	világos piros
151	1. szürke
152	2. szürke
153	világoszöld
154	világoskék
155	3. szürke

c) *Vezérlés POKE-utasítással*

A képernyő tartalmára vonatkozó információkat ebben az üzemmódban a képernyőtár őrzi (ld. 3.6 fejezet). Természetesen meg kell különböztetnie a normál és inverz jeleket, de nem kell rögzítenie a vezérlőjeleket, amik nem

jelennek meg a képernyőn. Ebben az esetben nem az ASCII-, hanem képernyőkódok kapnak szerepet. Amikor egy jelet közvetlenül a tárba akarun beírni, ezeket a kódokat használjuk. Pl. a

```
POKE 1024,1
```

utasítás hatására a képernyő bal felső sarkába egy A betű kerül, ami azonba egyelőre láthatatlan (hacsak előzőleg nem volt már ott valamilyen jel Hiányzik ugyanis a pontszín a színtár megfelelő helyéről. Vigyük be tehá a következő utasítássort:

```
POKE 55296,5
```

Most már látható az A betűnk, mégpedig zöld színben. Ezeket az összefüggés ket a 3.6.1 fejezetben részletesen ismertettük.

A következő oldalakon három különböző programot mutatunk be, amelye ugyanazt a képet eredményezik, de más-más módszerrel. Itt most tekintsün el attól, hogy a bemutatott eljárások nem a legcélszerűbbek. Csupán az lényeg, hogy összehasonlítsuk őket és a tapasztalatainkat majd a későbbiekbe hasznosítsuk.

```
1 REM *** P 5 ***
2 REM
100 REM ******************
110 REM *                *
120 REM * GRAFIKA / PRINT *
130 REM *                *
140 REM ******************
150 REM
160 PRINT CHR$(147) : PRINT : PRINT
165 REM KEPERNYOTORLES/SOREMELES
170 PRINT"                              ":REM MN
180 PRINT"                              ":REM MUIUK
190 PRINT"                              ":REM JKM
200 PRINT"                              ":REM NMUK
210 PRINT"                              ":REM P
220 PRINT"                              ":REM PPPLLPP
230 PRINT"                              ":REM NMM
240 PRINT"                              ":REM NML
250 PRINT"                              ":REM NML
260 PRINT"                              ":REM OYYYYPN
270 PRINT"                              ":REM HASASNN
280 PRINT"                              ":REM HZXZXNNASUI
290 PRINT"                              ":REM HOPNNUEEKJI
300 PRINT"                              ":REM H,ONNNEUCCWE
310 PRINT"                              ":REM YYOYYYOYYYYY
320 PRINT"                              ":REM LPPPPPPPP@
```



```
1 REM *** P 6 ***
2 REM
100 REM *****************
110 REM *               *
120 REM * GRAFIKA / CHR$ *
130 REM *               *
140 REM *****************
150 REM
160 PRINT CHR$(147):PRINT:PRINT
165 REM KEPERNYOTORLES/SOREMELES
170 FOR X=1 TO 290:REM ASCII-KODOK BETOLTESE
180 READ CH:REM DATA-SOROK OLVASASA
190 PRINT CHR$(CH);:REM JELEK IRASA
200 NEXT X
210 REM
220 REM *************
230 REM * DATA-SOROK *
240 REM *************
250 REM
260 DATA 32,  32, 205,  32, 206,  13
270 REM
280 DATA 32, 205, 213, 201,  32,  32
290 DATA 32,  32,  32,  32,  32,  32
300 DATA 32,  32,  32, 213, 203,  13
310 REM
320 DATA 32,  32, 202, 203, 205,  13
330 REM
340 DATA 32, 206,  32, 205,  32,  32
350 DATA 32,  32,  32,  32,  32,  32
360 DATA 32, 213, 203,  13
370 REM
380 DATA 32,  32,  32,  32,  32,  32
390 DATA 32,  32,  32,  32,  32,  32
400 DATA 32, 175,  13
410 REM
420 DATA 32,  32,  32,  32,  32,  32
430 DATA 32,  32,  32,  32, 175, 175
440 DATA175, 204, 204, 175, 175,  13
450 REM
460 DATA 32,  32,  32,  32,  32,  32
470 DATA 32,  32,  32, 206, 205,  32
480 DATA 32,  32,  32,  32,  32, 205
490 DATA 13
500 REM
510 DATA 32,  32,  32,  32,  32,  32
520 DATA 32,  32, 206,  32,  32, 205
530 DATA 32,  32,  32,  32,  32, 182
540 DATA 13
550 REM
560 DATA 32,  32,  32,  32,  32,  32
570 DATA 32, 206,  32,  32,  32,  32
580 DATA205,  32,  32,  32,  32, 182
590 DATA 13
600 REM
610 DATA 32,  32,  32,  32,  32,  32
```

```
620 DATA 32, 207, 183, 183, 183, 183
630 DATA208,  32,  32,  32,  32, 206
640 DATA 13
650 REM
660 DATA 32,  32,  32,  32,  32,  32
670 DATA 32, 165, 176, 174, 176, 174
680 DATA170,  32,  32,  32, 206,  13
690 REM
700 DATA 32,  32,  32,  32,  32,  32
710 DATA 32, 165, 173, 189, 173, 189
720 DATA170,  32,  32, 206,  32,  32
730 DATA176, 174, 213, 201,  13
740 REM
750 DATA 32,  32,  32,  32,  32,  32
760 DATA 32, 165,  32, 207, 208,  32
770 DATA170,  32, 206,  32,  32, 213
780 DATA177, 177, 203, 202, 201,  13
790 REM
800 DATA 32,  32,  32,  32,  32,  32
810 DATA 32, 165,  32, 207, 170,  32
820 DATA170, 206,  32,  32,  45, 177
830 DATA215, 195, 195, 215, 177,  13
840 REM
850 DATA 32,  32,  32,  32,  32,  32
860 DATA 32, 183, 183, 207, 183, 183
870 DATA183,  32,  32,  32, 207, 183
880 DATA183, 183, 183, 183, 183, 183
890 DATA183, 183,  13,
900 REM
910 DATA 32,  32,  32,  32,  32,  32
920 DATA 32,  32,  32, 204, 175, 175
930 DATA175, 175, 175, 175, 165,  13
```

```
1 REM *** P 7 ***
2 REM
100 REM ******************
110 REM *                *
120 REM * GRAFIKA / POKE *
130 REM *                *
140 REM ******************
150 REM
160 SZ=5:REM SZIN = ZOLD
170 PRINT CHR$(147):REM KEPERNYOTORLES
180 VR= 1024+2*40:REM POKE KEZDOCIM (KEPERNYOTAR.)
190 FR=55296+2*40:REM POKE KEZDOCIM (SZINTAROLO)
200 FOR Y=0 TO 15:REM 16 SOR
210 READ JS:REM JELEK SZAMA SORONKENT
220 VR=VR+40:REM KOVETKEZO SOR (+ 40 BYTE)
230 FR=FR+40:REM KOVETKEZO SOR (+ 40 BYTE)
240 FOR X=0 TO JS-1:REM JELEK BEIRASA A TAROLOBA
250 READ KK:REM KEPERNYOKODOK BEOLVASASA
260 POKE VR+X,KK:REM ES BEIRASA A KEPERNYOTAROLOBA
270 POKE FR+X,SZ:REM SZIN BEIRASA A SZINTAROLOBA
```

```
280 NEXT X
290 NEXT Y
300 REM
310 REM ****************
320 REM * KEPERNYOKODOK *
330 REM ****************
340 REM
350 DATA  5
360 DATA 32,  32,  77,  32,  78
370 REM
380 DATA 17
390 DATA 32,  77,  85,  73,  32,  32
400 DATA 32,  32,  32,  32,  32,  32
410 DATA 32,  32,  32,  85,  75
420 REM
430 DATA  5
440 DATA 32,  32,  74,  75,  77
450 REM
460 DATA 15
470 DATA 32,  78,  32,  77,  32,  32
480 DATA 32,  32,  32,  32,  32,  32
490 DATA 32,  85,  75
500 REM
510 DATA 14
520 DATA 32,  32,  32,  32,  32,  32
530 DATA 32,  32,  32,  32,  32,  32
540 DATA 32, 111
550 REM
560 DATA 17
570 DATA 32,  32,  32,  32,  32,  32
580 DATA 32,  32,  32,  32, 111, 111
590 DATA111,  76,  76, 111, 111
600 REM
610 DATA 18
620 DATA 32,  32,  32,  32,  32,  32
630 DATA 32,  32,  32,  78,  77,  32
640 DATA 32,  32,  32,  32,  32,  77
650 REM
660 DATA 18
670 DATA 32,  32,  32,  32,  32,  32
680 DATA 32,  32,  78,  32,  32,  77
690 DATA 32,  32,  32,  32,  32, 118
700 REM
710 DATA 18
720 DATA 32,  32,  32,  32,  32,  32
730 DATA 32,  78,  32,  32,  32,  32
740 DATA 77,  32,  32,  32,  32, 118
750 REM
760 DATA 18
770 DATA 32,  32,  32,  32,  32,  32
780 DATA 32,  79, 119, 119, 119, 119
790 DATA 80,  32,  32,  32,  32,  78
800 REM
810 DATA 17
820 DATA 32,  32,  32,  32,  32,  32
```

```
830 DATA 32, 101, 112, 110, 112, 110
840 DATA106, 32, 32, 32, 78
850 REM
860 DATA 22
870 DATA 32, 32, 32, 32, 32, 32
880 DATA 32, 101, 109, 125, 109, 125
890 DATA106, 32, 32, 78, 32, 32
900 DATA112, 110, 85, 73
910 REM
920 DATA 23
930 DATA 32, 32, 32, 32, 32, 32
940 DATA 32, 101, 32, 79, 80, 32
950 DATA106, 32, 78, 32, 32, 85
960 DATA113, 113, 75, 74, 73
970 REM
980 DATA 23
990 DATA 32, 32, 32, 32, 32, 32
1000 DATA 32, 101, 32, 79, 106, 32
1010 DATA106, 78, 32, 32, 45, 113
1020 DATA 87, 67, 67, 87, 113
1030 REM
1040 DATA 26
1050 DATA 32, 32, 32, 32, 32, 32
1060 DATA 32, 119, 119, 79, 119, 119
1070 DATA119, 32, 32, 32, 79, 119
1080 DATA119, 119, 119, 119, 119, 119
1090 DATA119, 119
1100 REM
1110 DATA 17
1120 DATA 32, 32, 32, 32, 32, 32
1130 DATA 32, 32, 32, 76, 111, 111
1140 DATA111, 111, 111, 111, 101
```

A három program közül az elsőben PRINT-utasítások sorozatával állítjuk elő képet. Esetünkben ez a leggyorsabb, legrövidebb és legkedvezőbb megoldás.

A kép készítése a 160-as sorban, képernyőtörléssel kezdődik: PRINT CHR$ (147). Maga a kép a grafikus jelkészlet elemeiből épül fel. A REM-es sorokban felsoroltuk a felhasznált billentyűket, amelyeket a SHIFT- vagy C=-billentyűkkel együtt kell lenyomni.

A P6 programban a képet szintén PRINT-utasításokkal állítjuk elő, de nem a billentyűk lenyomásával, hanem a jelek ASCII-kódjának felhasználásával. Ezeket a értékeket DATA-sorokban rögzítjük. Mint tudjuk, a DATA-sorokban különböző, egymástól vesszővel elválasztott adatokat tárolunk, amelyeket READ-utasítással olvasunk be a tárba. Esetünkben a beolvasás a 180-as sorban történik. A beolvasott érték a CH-változóba kerül, amihez a 190-es sor hozzárendeli a megfelelő jelet és kijelzi a képernyőre. Ez az egész folyamat egy FOR...NEXT-ciklus magját képezi, ami összesen 290-szer fut le, beolvasva és kijelezve mind a 290 jelet.

Minden képsor végén a 13-as ASCII-kód áll, ami a RETURN-t jelenti és új sor kezdését eredményezi. Az egyes képsorokat REM-mel választottuk el.

Bizonyára feltűnt, hogy milyen sok DATA-sorra van szükség. Emiatt ezt a módszert nem is lehet igazán hatékonyan alkalmazni. Valamelyest javulna a helyzet, ha a sorok elején levő üres helyeket (32) nem kellene egyenként megadnunk. Próbáljuk megoldani ezt a problémát!

A P7 programmal a POKE-utasítás alkalmazását, ill. a jeleknek közvetlenül a képernyőtárba való beírását szemléltetjük. Ebben az esetben a DATA-sorok a képernyőkódokat tartalmazzák. Nem elég azonban, ha csak a képernyőkódokat írjuk be, hanem ezenkívül a színtárban a megfelelő színt is jelenként rögzítenünk kell. A programban ezt két FOR...NEXT-ciklussal valósítottuk meg. A külső a képsor számát állítja elő (200–290-es sorok). Minden ciklusban először beolvassa a JS-változóba az adott sorba beírandó jelek számát (210. sor). A belső ciklus (240–280-as sorok) egyenként beolvassa a képernyőkódokat (KK), majd beírja a képernyőtár megfelelő tárrekeszébe (260-as sor). Ezután a színtár hozzá tartozó byte-jába beírja a színkódot (270-es sor). Ebben a példában is rövidíthetők a DATA-sorok.

Előbbi példáinkból meggyőződhettünk arról, hogy ugyanazt a grafikus képet több módon is előállíthatjuk. Hogy melyik a legjobb módszer, azt a jövőben mindenkinek önállóan kell eldöntenie, a kitűzött céltól függően. Általánosságban csak annyit mondhatunk, hogy egy igazán szép és aprólékosan kidolgozott grafika mindig sok munkát és időt igényel. Különösen, ha ehhez saját jelkészletet is használunk (ld. 4.4 fejezet).

Matematikai számításokon alapuló rajzokat, ábrákat is készíthetünk. A következő két program erre mutat példát:

```
1 REM *** P 8 ***
2 REM
100 REM ***************
110 REM *             *
120 REM * VELETLEN KEP *
130 REM *             *
140 REM ***************
150 REM
160 PRINT CHR$(147):REM KEPERNYOTORLES
165 REM VELETLEN ASCII-KODOK (177/178/179)
170 PRINT CHR$(RND(1)*3+177);
180 GOTO 170
```

Próbáljuk elemezni és megérteni a programot. Az RND (1) függvény 0 és 1 közötti véletlenszámokat állít elő.

A következő példában egy olyan alprogramot (rutint) mutatunk be, amellyel a kurzor a képernyő egész területére vonatkozóan vezérelhető. Standard BASIC-ben ez csak egy soron belül lehetséges. Az alprogram az 1000. sorban kezdődik:

```
1 REM *** P 9 ***
2 REM
100 REM ***************************
110 REM *                         *
120 REM * SZINUSZGORBE/SZOVEG UM. *
130 REM *                         *
140 REM ***************************
150 REM
160 PRINT CHR$(147)       :REM KEPERNYOTORLES
170 FOR X=0 TO 39
180 Y=13*SIN(X/3)+12      :REM FUGGVENY
190 GOSUB 1000:PRINT"*";:REM A JELEK ELHELYEZESE
200 NEXT X:END
210 REM
220 REM POZICIO SZAMITAS
230 REM ****************
1000 PRINTCHR$(19);:IFY>0THENFORZ=1TOY:PRINT:NEXTZ
1010 PRINT TAB(X);:RETURN
```

A "*"-jellel egy szinuszgörbét rajzoltunk (ld. 5.1 fejezet). Az 1000. sorban az X és Y paraméterek értékei az aktuális kurzorpozíció koordinátáit jelentik (X = oszlop, Y = sor).

A PRINT CHR$ (19) utasítást követően soremelések következnek, egészen a kívánt sor eléréséig. Az oszlopot a TAB-utasítással határozzuk meg.

Szöveg üzemmódban egészen jó grafikák készíthetők ezzel a rövid, de hatásos alprogrammal.

A képek előállításának másik módszere a képernyőre közvetlenül a billentyűzetről "rajzolni". Amikor a kép elkészült, minden sor elé beírjuk a sorszámot, egy PRINT-utasítást és a sorban levő jeleket idézőjelek közé tesszük. Vigyázzunk arra, hogy ne írjuk felül az ábrát, vagy ne toljuk el a sort. A PRINT-utasítást egy kérdőjellel (?) helyettesíthetjük és a sort a RETURN-billentyű lenyomásával fejezzük be. Ezzel a képernyő tartalmát rögzítettük. Van azonban néhány apróság, amire oda kell figyelnünk: Először is az, hogy minden sorban helyet kell hagynunk a sorszám, a kérdőjel és az idézőjelek számára. Ha ezekre a helyekre az ábrához is szükségünk van, akkor a programban ezt utólag kell kiegészíteni.

Ha a kurzorra végigmegyünk az utolsó képernyősoron, az egész képernyőtar-

talom egy sorral felfelé tolódik és a legfelső sor eltűnik. Ezenkívül egyik sorban sem használhatjuk az utolsó oszlopot, mert ilyenkor elmarad a soremelés és a következő sor folytatólagosan jelenik meg a képernyőn.

A PRINT-utasítássorban nem használhatjuk közvetlenül az inverz jeleket. Ha az ábra ilyet is tartalmaz, a következők szerint kell eljárnunk:

RVS ON-billentyű jel-billentyű RVS OFF-billentyű

"jel"-en itt a képernyőn inverz alakban megjelenítendő, de a programban normál alakban beírt jeleket értjük. A színeket sem adhatjuk meg közvetlenül, hanem csak a szín-vezérlőjeleket. Itt kell megjegyeznünk, hogy a vezérlőkódok csak akkor kerülnek be helyesen a PRINT-kifejezésbe, ha előtte lenyomtuk az idézőjel-billentyűt. (Máskülönben rögtön végrehajtásra kerülnek.) Utólagos beszúrásoknál ez nagyon zavaró lehet és nincs is rá szükség. Ugyanezt elérjük, ha az INST-billentyűvel teremtünk helyet a vezérlőjel számára. Ha például az "AB"-füzérben az A és B jelek közé egy vezérlőjelet akarunk beszúrni, akkor az A után lenyomjuk az INST-billentyűt és az így keletkezett üres helyre beírjuk a kívánt vezérlőjelet. Amint látjuk, ez nem túl kényelmes módja a grafika készítésének. Kezdetben, vagy alkalmanként ennek ellenére elfogadható. Ha azonban több és jobb minőségű képet akarunk készíteni, érdemes ehhez egy segédprogramot írni. Akit ez a téma bővebben is érdekel, annak a 4.4 fejezet áttanulmányozását ajánljuk.

4.2 A pontgrafika alkalmazása

Mind a többszínű, mind a nagyfelbontású grafika kezelése elég nehézkes. Ha semmi mást nem veszünk figyelembe, csak a rengeteg pont egy képzeletbeli koordináta-rendszerben való vezérlését, már akkor is egy csomó probléma merül fel. Mindez sok fejtörést és végül rengeteg számolást igényel. Ahhoz, hogy minden tényezőt figyelembe vehessünk, a grafikai munka egészét át kell látnunk. Egy egyszerű vonal, vagy még inkább egy kör megrajzolása is nehéz és komoly matematikai ismereteket feltételez, ezért csak gyakorlott programozóknak ajánljuk. Ebben a fejezetben különböző, más programokban is felhasználható rutinokat (alprogramokat) mutatunk be, amelyekkel a 3. fejezetben tárgyalt dolgokat a lehető legteljesebben kihasználhatjuk. Nem fontos a programok minden lépését megérteni, hiszen nem ezen, hanem az alkalmazáson van a hangsúly. Aki például nem tudja pontosan, mi az a szinusz vagy koszinusz, hagyja ki nyugodtan ezeket a részeket.

Mielőtt hozzáfognánk a programozáshoz, olvassuk át még egyszer a 3. fejezet ide vonatkozó részét, a 3.4 fejezetet. Az itt leírtak az érdeklődők számára fontos információkat tartalmaznak a rutinok átalakítására, más programokba való beillesztésére vonatkozóan.

Azzal tisztában kell lennünk, hogy egy BASIC-alprogram, amelynek áttekinthetőnek kell lennie, jóval lassabban dolgozik, mint a megfelelő gépi kódú rutin. A grafikus hatások legtöbbje BASIC-ben elég nehezen valósítható meg. Ha kipróbáljuk, hogy mennyi ideig tart például kört rajzolni, efelől semmi kétségünk sem marad. Emiatt a 4. fejezet végén közreadunk egy kis assemblerlistát (BASIC-betöltőprogramjával együtt), amelyben összegyűjtöttük a grafika leglényegesebb elemeit. Az egyes funkciók, mint kisebb BASIC-bővítők, BASIC-ből vezérelhetők.

Akik csak BASIC-ben akarnak programozni, azok számára a Függelékben összefoglaltuk azokat a szempontokat, amelyek figyelembevételével a program a lehető legkedvezőbben állítható össze.

Minden programnál abból kell kiindulni, hogy a grafikustár a $2000–$3FFF (8192–16383), a képernyőtár pedig a $0400–$07FF (1024–2047) címtartományban helyezkedik el.

Ez elvileg lehetetlenné teszi ugyan a szöveg és grafikus üzemmód egyidejű alkalmazását, de programtechnikailag mégis megoldható. Arra kell vigyáznunk, hogy a hosszú és/vagy sok tárhelyet igénylő programok ne ütközzenek a grafikai oldallal. Ha ez mégis előfordulna, a program első sorába beírt

```
POKE 45,0 : POKE 46,64
```

utasításokkal emeljük fel a változótartomány kezdőcímét $4000 (16384)-re.

Ügyeljünk arra, hogy minden programmódosításkor tönkremegy a grafikai oldal, és ha a programot tároljuk, a grafika is hibásan kerül a lemezre. Amikor tehát változtatni akarunk a programon, újra töltsük be, hajtsuk végre a kívánt módosításokat, majd rögtön, még a program elindítása előtt, tároljuk. Csak ezután szabad a módosított programot kipróbálni.

4.2.1 A grafika előkészítése

Mielőtt a képernyőre vinnénk egy figurát, természetesen gondoskodnunk kell arról, hogy az látható is legyen. Mindenekelőtt töröljük a képernyőt, hogy a nem odatartozó jelek ne zavarják a munkát.

A végén ismét törölnünk kell a képernyőt és a színeket, majd vissza kell állítanunk az eredeti szöveg üzemmódot. Összesen tehát négy programrészre, más szóval alprogramra (rutinra) van szükségünk:

- a grafika bekapcsolása,
- a grafikustár törlése,
- a színtár törlése,
- a grafika kikapcsolása.

Ezzel a négy művelettel a következőkben külön-külön foglalkozunk. A későbbiek során alprogramokként minden programunkban szerepelni fognak.

4.2.1.1 A grafika bekapcsolása

Nos, kezdhetjük. Nézzük, mire van szükség a grafika bekapcsolásához:

a) *A tár felosztása*

Először is azt kell eldöntenünk, hogy a 64-es tárterületét hogy osztjuk fel az egyes tárak (grafikustár, képernyőtár... stb.) között. Magát a felosztást a 24-es VIC-regiszter 3,4–7, valamint a CIA2 0. regiszterének 0/1 bitjeinek használatával végezzük el. Ezzel bővebben a 3.3 fejezetben foglalkoztunk.

A grafikustár eredeti helye a $2000–$3FFF (8192–16383), a képernyőtáré pedig a $0400–$07FF (1024–2047) címtartomány. Ha a grafikánkat más tártartományban akarjuk rögzíteni, el kell végeznünk a megfelelő módosításokat.

b) *Grafikai változatok*

El kell döntenünk, hogy többszínű, de kisebb felbontású, vagy egyszínű, de nagyobb felbontású grafikát akarunk-e készíteni. A két eljárás lényegét a 3.4 fejezetben ismertettük.

Foglalkozzunk most csak a nagyfelbontású grafikával. Ezt az üzemmódot a 17-es VIC-regiszter 5. bitjének bekapcsolásával és a 22-es VIC-regiszter 4. bitjének egyidejű törlésével választjuk ki. (A 22-es VIC-regiszter 4. bitje

eredetileg 0, ezért ha előzőleg nem kapcsoltuk be, nem kell vele törődnünk.) A mintapéldában ez két egyszerű POKE-utasítással történik (10070-es és 10080-as sorok).

A címtartomány kiválasztása szintén nagyon egyszerű. Tudjuk, hogy a VIC összesen 16 k címzésére képes. Alapállapotban az alsó 16 k-t éri el. Ebből most nem akarunk kilépni, ezért a CIA2 regiszterének tartalmát nem kell módosítanunk, csupán a grafikustár kezdőcímét kell a 24-es regiszter 3. bitjének bekapcsolásával a 16 k-os tartomány felső felébe helyezni. A következő alprogramban ez az 10090-es sorban történik:

```
1 REM *** P 10 ***
2 REM
10000 REM ***************************
10010 REM *                         *
10020 REM * A GRAFIKA BEKAPCSOLASA *
10030 REM *                         *
10040 REM ***************************
10050 REM
10060 V=53248:REM A VIC KEZDOCIME
10070 POKE V+17,PEEK(V+17) OR (8+3)*16
10075 REM GRAFIKA BEKAPCSOLASA
10080 POKE V+22, PEEK(V+22) AND 255-16
10085 REM TOBBSZINU UM. KIKAPCSOLASA
10090 POKE V+24, PEEK(V+24) OR 8
10095 REM GRAFIKUS TAR A $2000(8192)-RE
```

Az AND- és OR-utasításokat 2. fejezetben ismertettük. A sorszámozást azért választottuk ilyen különlegesen magasra, hogy minden további nélkül hozzáfűzhető legyen más programokhoz.

A szöveg üzemmódot a RUN-STOP/RESTORE billentyűkombinációval állíthatjuk vissza.

4.2.1.2 A grafikustár törlése

A grafikustár számára lefoglalt tártartomány előzőleg is tartalmazott valamilyen információkat, amelyek a grafika bekapcsolásakor pontok és vonalak összevisszaságaként fog jelentkezni a képernyőn. Ahhoz, hogy a képernyőt a grafikánk számára szabaddá tegyük, a grafikustár minden bitjét törölnünk kell. A POKE-utasítással viszonylag szerencsés helyzetben vagyunk, mert segítségével egyszerre 8 bit, ill. pont törölhető. Elegendő egy FOR...NEXT-

ciklus, ami a grafikustárat a kezdőcímtől ($2000=8192) az utolsó tárhelyig ($3FFF=16383) byte-onként nullázza. Mivel egy kép 320×200 = 64000 pontból áll, valójában 64000:8 = 8000 byte törléséről van szó.

```
1 REM *** P 11 ***
2 REM
10200 REM *************************
10210 REM *                       *
10220 REM * A GRAFIKUS TAR TORLESE *
10230 REM *                       *
10240 REM *************************
10250 REM
10260 GT=8192:REM GRAFIKUS TAR KEZDOCIME
10270 FOR X=GT TO GT+8000:REM 8000 BYTE
10280 POKE X,0:REM TORLES
10290 NEXT X
```

Ha eleindítjuk a programot, tapasztalni fogjuk, hogy ez bizony nagyon hosszadalmas művelet. A fejezet végén található assembler-program ugyanezt szinte pillanatok alatt elvégzi.

A GT-változó nem tartozik szorosan a programhoz, de általa a grafikustár kezdőcíme tetszés szerint megadható.

4.2.1.3 A színek törlése

Nagyfelbontású grafika esetében a színt a képernyőtár tartalmazza (3.4 fejezet). Ebben minden byte felső négy bitje adja az adott pont színét, míg az alsó négy bit a háttérszínt. A képernyőtár eredetileg a szövegképernyő jeleit tárolja, amik a grafikus üzemmód bekapcsolása után, kis színes négyzetek formájában, a képernyőn maradnak. Emiatt a grafika számára be kell állítani a képernyő minden pontjára a megfelelő színeket. Erre alkalmas a következő alprogram:

```
1 REM *** P 12 ***
2 REM
10400 REM *******************
10410 REM *                 *
10420 REM * A SZINEK TORLESE *
10430 REM *                 *
10440 REM *******************
10450 REM
10460 KT=1024:REM A KEPERNYOTAROLO KEZDOCIME
10470 SZ=6*16+7:REM PONT=KEK HATTER=SARGA
10480 FOR I=KT TO KT+1000:REM 1000 BYTE
10490 POKE I,SZ:REM A SZINKODOK BEIRASA
10500 NEXT I
```

A KT-változóban adjuk meg a képernyőtár aktuális kezdőcímét. Ezt az értéket az alprogram egyébként a főprogramból venné át. Ugyanez vonatkozik az SZ-változó értékére is, ami a byte-okba beírandó pont- és háttérszín színkódját adja.

Próbálkozhatunk a program változtatásával, de legyünk óvatosak, mert a számítógép "szívében" vagyunk. Ha pl. a KT-változónak 0 értéket adunk, azonnal el is búcsúzik tőlünk, mert ezzel közvetlenül a nullás lapba írunk, ahol az operációs rendszer működéséhez szükséges adatok tárolása történik.

4.2.1.4 A grafika kikapcsolása

Eddig erre csak egy módszert ismertünk, a RUN-STOP/RESTORE billentyűk egyidejű lenyomását.

Ez a program kényszerű befejezését jelenti és nem túl elegáns megoldás. Nézzük tehát, mit kell tennünk ahhoz, hogy a programunkat szépen, szabályosan fejezzük be. Emlékezzünk vissza, milyen teendőink voltak a bekapcsoláskor. Most ezek fordítottját kell végrehajtanunk, azaz:

- a 17-es VIC-regiszter 5. bitjének,
- a 22-es VIC-regiszter 4. bitjének (TSZ üzemmód esetén),
- a 24-es VIC-regiszter 3. bitjének törlése.

A megfelelő alprogram a következő:

```
1 REM *** P 13 ***
2 REM
10600 REM ***************************
10610 REM *                         *
10620 REM * A GRAFIKA KIKAPCSOLASA *
10630 REM *                         *
10640 REM ***************************
10650 REM
10660 V=53248:REM VIC KEZDOCIME
10670 POKE V+17,PEEK(V+17) AND 255-2*16
10675 REM GRAFIKA KIKAPCSOLASA
10680 POKE V+22,PEEK(V+22) AND 255-16
10685 REM TOBBSZINU UM. KIKAPCSOLASA
10690 POKE V+24,PEEK(V+24) AND 255-8
10695 REM JELKESZLETTAR.VISSZA $1000-RE
```

Ezzel minden együtt van a grafikai munka előkészítéséhez és befejezéséhez.

Mivel az ismertetett négy program közül csak a P10-es és a P12-es futása követhető a képernyőn, fűzzük össze egy programmá:

```
1 REM *** P 14 ***
2 REM
10000 REM *************************
10010 REM *                       *
10020 REM * A GRAFIKA BEKAPCSOLASA *
10030 REM *                       *
10040 REM *************************
10050 REM
10060 V=53248:REM A VIC KEZDOCIME
10070 POKE V+17,PEEK(V+17) OR (8+3)*16
10075 REM GRAFIKA BEKAPCSOLASA
10080 POKE V+22, PEEK(V+22) AND 255-16
10085 REM TOBBSZINU UM. KIKAPCSOLASA
10090 POKE V+24, PEEK(V+24) OR 8
10095 REM GRAFIKUS TAR A $2000(8192)-RE
10200 REM *********************
10210 REM *                   *
10220 REM * GR. TAROLO TORLESE *
10230 REM *          .        *
10240 REM *********************
10250 REM
10260 GT=8192:REM A GRAFIKUS TAROLO KEZDOCIME
10270 FOR I=GT TO GT+8000:REM 8000 BYTE
10280 POKE I,0:REM TORLES
10290 NEXT I
10400 REM *******************
10410 REM *                 *
10420 REM * A SZINEK TORLESE *
10430 REM *                 *
10440 REM *******************
10450 REM
10460 KT=1024:REM A KEPERNYOTAROLO KEZDOCIME
10470 SZ=6*16+7:REM PONT=KEK HATTER=SARGA
10480 FOR I=KT TO KT+1000:REM 1000 BYTE
10490 POKE I,SZ:REM A SZINKODOK BEIRASA
10500 NEXT I
10600 REM *************************
10610 REM *                       *
10620 REM * A GRAFIKA KIKAPCSOLASA *
10630 REM *                       *
10640 REM *************************
10650 REM
10660 V=53248:REM A VIC KEZDOCIME
10670 POKE V+17,PEEK(V+17) AND 255-2*16
10675 REM GRAFIKA KIKAPCSOLASA
10680 POKE V+22,PEEK(V+22) AND 255-16
10685 REM TOBBSZINU UM. KIKAPCSOLASA
10690 POKE V+24,PEEK(V+24) AND 255-8
10695 REM JELKESZLETTAR.VISSZA $1000-RE
```

RUN-nal elindítva mind a négy alprogram működése figyelemmel kísérhető a képernyőn.

Most térjünk rá azokra a bonyolultabb összefüggésekre, amelyekkel valamiféle képet is előállíthatunk.

4.2.2 Egyszerű alakzatok a grafikában

Miután a grafikus üzemmódot be és ki tudjuk kapcsolni, megpróbálkozhatunk a programozással. Az egyszerű pont megjelenítésével kezdjük, majd a geometriai alapelemekkel (vonal, kör) folytatva végül eljutunk az összetett alakzatok ábrázolásáig.

4.2.2.1 A pont

Az előzőekben már említettük, hogy a grafikai munkához a képernyőt az ún. grafikus koordináta-rendszerben képzeljük el. Egy pont helyzetét két koordinátával határozzuk meg. Az első az X koordináta (0–319), amely azt adja meg, hogy a kijelölt pont a képernyő bal szélétől hányadik a sorban. Az Y koordináta (0–199) ugyanez, de a képernyő felső szélétől számítva.

A koordináta-rendszer kezdőpontja (X=0; Y=0) a képernyő bal felső sarkában van. A jobb alsó sarok koordinátái: X=319, Y=199.

Ebben tehát megállapodtunk. Igen ám, de a számítógéppel nem tudjuk közvetlenül közölni a megfelelő koordinátákat. Ha az erre vonatkozó fejezetet (3.4.2) elolvastuk, tudjuk, hogy a grafikustár nem csupán egyszerű leképezése a grafikus képernyőnek. Ahhoz, hogy a koordinátáival meghatározott pont információit valóban a neki megfelelő byte-ba, ill. bitbe írhassuk, előzőleg néhány átszámítást kell végezzünk.

Nézzük először az Y koordinátát:

Tudjuk, hogy egy képernyősor (amibe szöveg üzemmódban írni tudunk) 8 egymás alatti pontsorból áll. Ezért tehát először is azt kell meghatároznunk, hogy az általunk kijelölt pont melyik képernyősorban van. Ezt úgy kapjuk meg, ha az Y koordináta értékét elosztjuk 8-cal és vesszük az eredmény egész részét:

a képernyősor száma = INT (Y/8).

Mivel minden képernyősor 40*8 = 320 byte-ból áll, az így kapott értéket 320-szal megszorozva megkapjuk az illető sor táron belüli relatív kezdőcímét:

a képernyősor relatív kezdőcíme = 320*INT (Y/8).

Az előbbi osztás maradéka jelenti a pontsor képernyősoron belüli számát, amit a relatív kezdőcím értékéhez hozzá kell adnunk:

a pontsor relatív kezdőcíme = 320*INT (Y/8)+(Y AND 7).

Az X koordináták esetében egy kicsit bonyolultabb a helyzet, mert nem csak a byte-okat, hanem a biteket is meg kell különböztetnünk.

Először is át kell számítanunk a megfelelő byte címét a pontsor relatív kezdőcíméhez viszonyítva:

byte relatív címe = 8*INT (X/8).

Ezután kiszámítjuk az adott bit byte-on belüli helyét. Erre egy ún. számítási maszkot használhatunk, amelyben az illető bit értéke 1, minden többi bit értéke pedig 0.

maszk = 2 ↑ (7 -(X AND 7))

Ezeket a részeket az alábbiakban egy programmá fűztük össze:

```
1 REM *** P 15 ***
2 REM
10700 REM **********************
10710 REM *                    *
10720 REM * PONT BEKAPCSOLASA *
10730 REM *                    *
10740 REM **********************
10750 REM
10760 RA=320*INT(Y/8) + (Y AND 7)
10770 BA=8*INT(X/8)
10780 MA=2↑(7-(X AND 7))
10790 AD=GT+RA+BA
10800 POKE AD,PEEK(AD) OR MA
10810 REM
10900 REM ****************
10910 REM *              *
10920 REM * PONT TORLESE *
10930 REM *              *
10940 REM ****************
10950 REM
```

```
10960 RA=320*INT(Y/8) + (Y AND 7)
10970 BA=8*INT(X/8)
10980 MA=255-2↑(7-(X AND 7))
10990 AD=GT+RA+BA
11000 POKE AD,PEEK(AD) AND MA
11010 REM
11020 REM BELSO PARAMETEREK
11030 REM ****************
11040 REM RA=A PONTSOR RELATIV KEZDOCIME
11050 REM BA=A BYTE CIME A PONTSORON BELUL
11060 REM MA=MASZK
11070 REM AD=A BYTE ABSZOLUT CIME
11080 REM
11090 REM ATADANDO PARAMETEREK
11100 REM ********************
11110 REM GT=GRAFIKUS TAROLO KEZDOCIME (PL.8192)
11120 REM X=X-KOORDINATA
11130 REM Y=Y-KOORDINATA
```

Láthatjuk, hogy különbség van a pont bekapcsolása és törlése között, ezért ezeket az eseteket ténylegesen is külön kezeljük.

A programban a GT-változó a grafikustár kezdőcímét tartalmazza, ami esetünkben 8192. Természetesen ez a két rutin is felhasználható grafikus programjainkban alprogramként, de ilyenkor RETURN-nel kell befejeznünk őket. A következő programban erre is látunk példát:

```
1 REM *** P 16 ***
2 REM
100 REM ****************
110 REM *              *
120 REM * SZINUSZGORBE *
130 REM *              *
140 REM ****************
150 REM
160 V=53248:REM VIC KEZDOCIME
170 GT=8192:REM GRAFIKUS TAR. KEZDOCIME
175 POKE V+32,10:REM KERETSZIN
180 GOSUB 10000:REM GRAFIKA BEKAPCSOLASA3
190 GOSUB 10200:REM GRAFIKA TORLESE
200 SZ=7*16+2:GOSUB 10400:REM SZINEK BEALLITASA
210 Y=100:REM X-TENGELY RAJZOLASA
220 FOR X=0 TO 319
230 GOSUB 10700:REM PONT RAJZOLASA
240 NEXT X
250 X=160:REM Y-TENGELY RAJZOLASA
260 FOR Y=0 TO 199
270 GOSUB 10700:REM PONT RAJZOLASA
280 NEXT Y
290 FOR X=0 TO 319:REM SZINUSZGORBE RAJZOLASA
300 Y=70*SIN(X/25.5)+99
```

```
310 GOSUB 10700:REM PONT RAJZOLASA
320 NEXT X
330 POKE 198,0:REM BILLENTYU TORLESE
340 WAIT 198,255:REM VARAKOZAS EGY BILLENTYURE
350 GOSUB 10600:REM GRAFIKA KIKAPCSOLASA
360 END
370 REM
10000 REM ************************
10020 REM * GRAFIKA BEKAPCSOLASA *
10040 REM ************************
10050 REM
10070 POKE V+17,PEEK(V+17) OR (8+3)*16
10075 REM GRAFIKA BEKAPCSOLASA
10080 POKE V+22,PEEK(V+22) AND 255-16
10085 REM TOBBSZINU UM. KIKAPSOLASA
10090 POKE V+24,PEEK(V+24) OR 8
10095 REM GRAFIKUS TAR. $2000(8192)-RE
10100 RETURN
10110 REM
10200 REM **********************
10220 REM * GR. TAROLO TORLESE *
10240 REM **********************
10250 REM
10270 FOR I=GT TO GT+8000:POKE I,0:NEXT I
10300 RETURN
10310 REM
10400 REM ********************
10410 REM * A SZINEK TORLESE *
10420 REM ********************
10450 REM
10460 KT=1024:REM KEPERNYOTAROLO KEZDOCIME
10480 FOR I=KT TO KT+1000:POKE I,SZ:NEXT I
10510 RETURN
10520 REM
10600 REM ************************
10620 REM * GRAFIKA KIKAPCSOLASA *
10640 REM ************************
10650 REM
10670 POKE V+17,PEEK(V+17) AND 255-6*16
10675 REM GRAFIKA KIKAPCSOLASA
10680 POKE V+22,PEEK(V+22) AND 255-16
10685 REM TOBSZINU UM. KIKAPCSOLASA
10690 POKE V+24,PEEK(V+24) AND 255-8
10695 REM JELKESZLETTAR. VISSZA $1000-RE
10700 REM ********************
10720 REM * PONT ELHELYEZESE *
10740 REM ********************
10750 REM
10760 RA=320*INT(Y/8)+(Y AND 7)
10770 BA=8*INT(X/8)
10780 MA=2↑(7-(X AND 7))
10790 AD=GT+RA+BA
10800 POKE AD,PEEK(AD) OR MA
10810 RETURN
```

A grafikustár törlését végző programrészben (10200–10300-as sorok) elhagytuk a REM-es sorokat, hogy meggyorsítsuk egy kicsit a programfutást. A felhasznált rutinok teljesen megegyeznek a P10, P11, P12 és P13-as programokkal.

Jobban megértjük a program működését, ha más értékekkel is kipróbáljuk. Változtassuk meg például a 300-as sorban megadott paramétereket!

Csak gyakorlással, kísérletezéssel jutunk előbbre.

4.2.2.2 A vonal

Valamivel nehezebb feladat a képernyő két pontja közé egy egyenest rajzolni. Szinte naponta találkozunk vele, de talán még sosem gondoltunk arra, hogy megvalósítása milyen összetett probléma. Hogyan állapítjuk meg például azt, hogy a képernyő mely pontjai alkotják a kérdéses egyenest? Ehhez meg kell ismerkednünk egy kicsit az analitikus geometriával. Ne ijedjünk meg! Ez az ijesztő kifejezés nagyon ártatlan dolgot takar.

Akit ezek a dolgok nem érdekelnek, nyugodtan át is ugorhatja a következő néhány sort, mert úgyis csak arra a képletre lesz szüksége, amit a sok összefüggésből végül előállítunk az egyenest alkotó pontok helyzetének kiszámításához.

Arról már bizonyára mindenki hallott, hogy a koordináta-rendszerben elhelyezett egyenest egy elsőfokú egyenlettel jellemezhetjük:

$$y = mx + n$$

ahol x és y az egyenes egy pontjának koordinátái, m az egyenes meredeksége, n pedig az egyenes és az y tengely metszéspontja. A képlet egyszerű átalakításával kifejezhetjük n-t:

$$n = y - mx$$

Ha ismerjük az egyenes két végpontját – P1(x1, y1) és P2(x2, y2) –, akkor két képletet írhatunk fel:

$$n = y1 - mx1,$$

$$n = y2 - mx.2$$

Mivel ugyanazon egyenes két pontjáról van szó, a két képletből kifejezhetjük az m értékét:

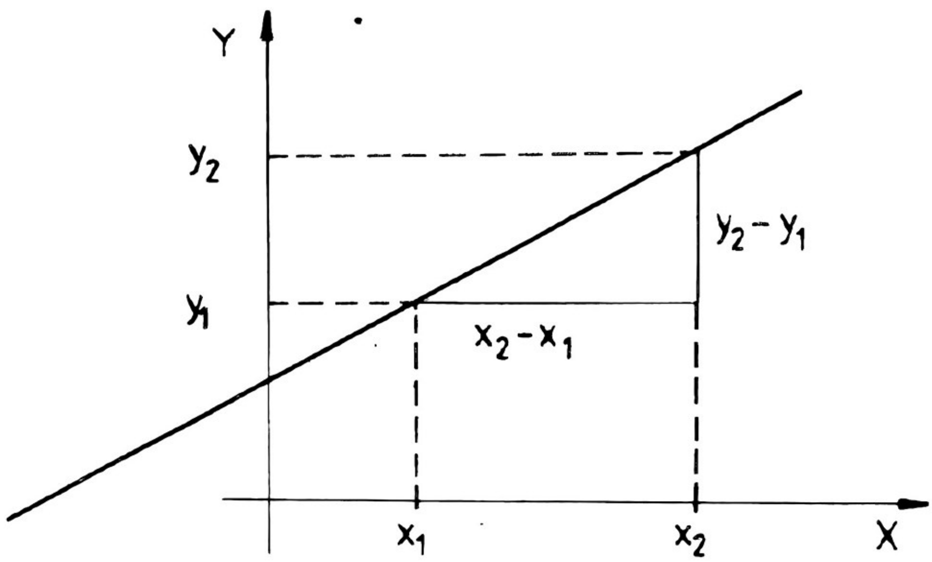
$$m = \frac{y2 - y1}{x2 - x1}$$

n=0 esetén az egyenes áthalad a koordináta-rendszer kezdőpontján (0, 0 pont).

Ha az egyenes általános egyenletébe behelyettesítjük a két végpont koordinátáival kifejezett m és n paramétereket, majd elvégezzük a lehetséges összevonásokat, az alábbi összefüggést kapjuk:

$$y = \frac{y2 - y1}{x2 - x1} * (x - x2) + y2.$$

Ez az alapja a mintaprogramunknak (P17). A 10970-es, 11000-es és 11020-as sorokban alkalmazzuk, ahol egy FOR...NEXT-ciklussal minden X koordinátához kiszámítjuk a megfelelő Y koordinátát.



Egyedül akkor merül fel probléma, ha az egyenes függőleges. Ekkor ugyanis

$$x2 = x1 \longrightarrow x2 - x1 = 0,$$

és a számítógép

DIVISION BY ZERO

hibaüzenettel figyelmeztetne arra, hogy 0-val akarunk osztani. Ezt úgy kerültük el, hogy a 10990-es sorban elágaztatjuk a programot és a függőlegest közvetlenül megrajzoljuk.

Nézzük a programot:

```
1 REM *** P 17 ***
2 REM
100 REM ***********
110 REM *         *
120 REM * EGYENES *
130 REM *         *
140 REM ***********
150 REM
160 V=53248 : GT=8192
170 GOSUB 10000:REM GRAFIKA BEKAPCSOLASA
180 SZ=1*16+0:GOSUB 10400:REM SZINEK BEALLITASA
190 GOSUB 10200:REM GRAFIKUS TAR. TORLESE
270 X1=110:Y1=120:X2=140:Y2=150
275 REM A VEGPONTOK KOORDINATAI
280 GOSUB 10900:REM EGYENES RAJZOLASA
290 WAIT 198,255:REM VARAKOZAS EGY BILLENTYURE
300 GOSUB 10600:REM GRAFIKA KIKAPCSOLASA
310 END
320 REM
10000 REM *************************
10020 REM * GRAFIKA BEKAPCSOLASA *
10040 REM *************************
10050 REM
10070 POKE V+17,PEEK(V+17) OR (8+3)*16
10075 REM GRAFIKA BEKAPCSOLASA
10080 POKE V+22,PEEK(V+22) AND 255-16
10085 REM TOBBSZINU UM. KIKAPCSOLASA
10090 POKE V+24,PEEK(V+24) OR 8
10095 REM GRAFIKUS TAR. $2000(8192)-RE
10100 RETURN
10110 REM
10200 REM **********************
10220 REM * GR. TAROLO TORLESE *
10240 REM **********************
10250 REM
10270 FOR I=GT TO GT+8000:POKEI,0:NEXT I
10300 RETURN
10310 REM
10400 REM ******************
10420 REM * SZINEK TORLESE *
10440 REM ******************
10450 REM
10460 KT=1024:REM A KEPERNYOTAROLO KEZDOCIME
10480 FOR I=KT TO KT+1000:POKEI,SZ:NEXT I
10510 RETURN
10520 REM
```

```
10600 REM ************************
10620 REM * GRAFIKA KIKAPCSOLASA *
10640 REM ************************
10650 REM
10670 POKE V+17,PEEK(V+17) AND 255-6*16
10675 REM GRAFIKA KIKAPCSOLASA
10680 POKE V+22,PEEK(V+22) AND 255-16
10685 REM TOBBSZINU UM. KIKAPCSOLASA
10690 POKE V+24,PEEK(V+24) AND 255-8
10695 REM JELKESZLETTAR. VISSZA $1000-RE
10696 RETURN
10700 REM ********************
10720 REM * PONTOK SZAMITASA *
10740 REM ********************
10750 REM
10760 RA=320*INT(Y/8)+(Y AND 7)
10770 BA=8*INT(X/8)
10780 MA=2↑(7-(X AND 7))
10790 AD=GT+RA+BA
10800 POKE AD,PEEK(AD) OR MA
10810 RETURN
10900 REM
10910 REM *********************
10930 REM * EGYENES RAJZOLASA *
10950 REM *********************
10960 REM
10970 DY=Y2-Y1:DX=X2-X1:REM•KULONBSEGEK
10980 Y=Y2:X=X2:REM Y-START
10990 IF DX=0 THEN FOR Y=Y2 TO Y1 STEP SGN(-DY):GOSUB 11060:NEXT
Y:GOTO 11050
10995 REM FUGGOLEGES
11000 DD=DY/DX:REM AZ EGYENES EMELKEDESE
11010 FOR X=X2 TO X1 STEP SGN(-DX)
11020 Z=INT(DD*(X-X2)+Y2):REM AZ EGYENES EGYENLETE
11030 IF Z<>Y THEN Y=Y+SGN(-DY):GOSUB 11060:GOTO 11030
11035 REM FUGGOLEGES RAJZOLASA
11040 GOSUB 11060:NEXT X:REM KOVETKEZO X-KOORD.
11050 RETURN
11060 GOSUB 10760:X=X+1:GOSUB 10760:X=X-1:RETURN
11065 REM DUPLA SZELESSEG RAJZOLASA
```

Mint már annyiszor, most is tapasztalhatjuk, hogy a BASIC-program sebessége messze elmarad gépi nyelvű megfelelőjének sebességétől. Ettől függetlenül ez a rutin alprogramként jó szolgálatot tehet.

Ha nem akarjuk kivárni a grafikustár szabályos törlését, írjunk a 190-es sor elé egy REM-et.

A programban használt új változók jelentése:

X1/Y1 és X2/Y2:	az egyenes két végpontjának koordinátái,
DX/DY:	a koordinátapárok különbsége,

DD: az egyenes meredeksége (m),
X/Y: az egyenes egy-egy pontjának koordinátái,
Z: közbenső tár.

Két dologra kell még kitérnünk. Az egyik az SGN-függvény, a másik a 11060-as sor.

Az SGN-függvénynek nagyon hasznos feladata van: ha az argumentum (a zárójelben levő kifejezés) értéke pozitív, a függvény értéke: +1, ha negatív: –1, ha 0, akkor a függvény értéke is 0. Ezek alapján az SGN-függvény számok, kifejezések előjelének meghatározására alkalmas.

A 11060-as sorban minden pontot megduplázhatunk. Erre azért van szükség, mert ha egy pontnak X irányban nincs szomszédja, csak nagyon gyengén, vagy egyáltalán nem látható.

Sok sikert kívánunk a kísérletezéshez!

4.2.2.3 Az ellipszis és a kör

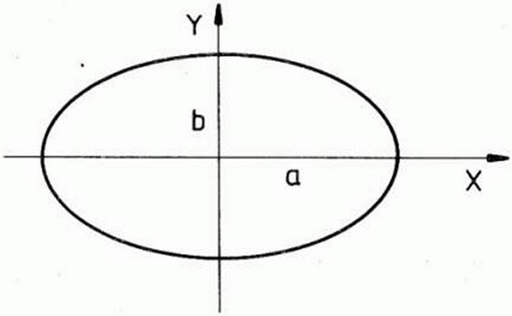
Egy másik gyakran alkalmazott grafikai elem a kör, vagy általánosságban az ellipszis. Ezek rajzolására is bemutatunk néhány szemléltető programot, de először néhány szóban vázoljuk a szükséges matematikai alapismereteket.

A továbbiakban az ellipszis rajzolásának két módját mutatjuk be és itt a könnyebben érthetőt tárgyaljuk. A másik módszerre, ami az ún. polárkoordinátákon alapul és körívek rajzolására is alkalmas, az 5. fejezet Kördiagramok című részében térünk ki.

Nézzük tehát a matematikai alapismereteket. Első példánkban az ellipszis középponti egyenletéből indulunk ki:

$$\frac{x^2}{a} + \frac{y^2}{b} = 1,$$

ahol x és y az ellipszis egy pontjának koordinátái, a és b pedig az ellipszis sugarai. Az ellipszis középpontja esetünkben a koordináta-rendszer kezdőpontjában (0, 0 pont) van.



Ahhoz, hogy ezt az egyenletet a programban felhasználhassuk, ki kell fejeznünk belőle y-t:

$$y = b * \sqrt{1 - \frac{x^2}{a^2}}$$

Ezzel a komplikáltnak tűnő egyenlettel számíthatjuk ki az ellipszis pontjainak koordinátáit. Figyelembe kell vennünk, hogy egyszerre mindig csak egy 90 fokos ív ábrázolható, mert az egyenlet tulajdonképpen nem függvényt ábrázol. A többi ívet úgy kapjuk meg, hogy ezt a megrajzolt ívet megfelelően tükrözzük. A gyakorlatban ez úgy valósul meg, hogy módosítjuk az x és y paraméterek előjelét.

A programban ez X-re vonatkozóan a 11150–11190-es sorokban egy FOR... NEXT-ciklussal történik úgy, hogy F2 értéke egymás után +1 és –1 értéket vesz fel.

A megfelelő Y koordinátákat a 11180-as sorban számítjuk ki.

A fenti egyenlet azokra az ellipszisekre vonatkozik, amelyeknek a koordinátarendszer kezdőpontjában van a középpontjuk. Ettől eltérő esetben az X és Y értékekhez előjelhelyesen hozzá kell adni a középpont X és Y koordinátáinak értékét.

Ha kört akarunk rajzolni, akkor a és b értékét egyenlőnek kell választanunk, mivel a kör egy olyan speciális ellipszis, amelynek a sugarai egyenlők.

```
1 REM *** P 18 ***
2 REM
100 REM *************
110 REM *           *
120 REM * ELLIPSZIS *
130 REM *           *
140 REM *************
150 REM
```

```
160 V=53248 : GT=8192
170 GOSUB 10000:REM GRAFIKA BE
180 SZ=1*16+0:GOSUB 10400:REM SZINEK BEALLITASA
190 GOSUB 10200:REM GRAFIKUS TAR. TORLESE
270 XR=40 :YR=20 :X0=160 :Y0=100
275 REM XR/YR=SUGARAK;X0/Y0=A KOZEPPONT KOORD.-I
280 GOSUB 11100:REM ELLIPSZIS RAJZOLASA
290 WAIT 198,255:REM VARAKOZAS EGY BILLENTYURE
300 GOSUB 10600:REM GRAFIKA KI
310 END
320 REM
10000 REM ************************
10020 REM * GRAFIKA BEKAPCSOLASA *
10040 REM ************************
10050 REM
10070 POKE V+17,PEEK(V+17) OR (8+3)*16
10075 REM GRAFIKA BEKAPCSOLASA
10080 POKE V+22,PEEK(V+22) AND 255-16
10085 REM TOBBSZINU UM. KIKAPCSOLASA
10090 POKE V+24,PEEK(V+24) OR 8
10095 REM GRAFIKUS TAR. $2000(8192)-RE
10100 RETURN
10110 REM
10200 REM **********************
10220 REM * GR. TAROLO TORLESE *
10240 REM **********************
10250 REM
10270 FOR I=GT TO GT+8000:POKEI,0:NEXT I
10300 RETURN
10310 REM
10400 REM ******************
10420 REM * SZINEK TORLESE *
10440 REM ******************
10450 REM
10460 KT=1024:REM A KEPERNYOTAROLO KEZDOCIME
10480 FOR I=KT TO KT+1000:POKEI,SZ:NEXT I
10510 RETURN
10520 REM
10600 REM ************************
10620 REM * GRAFIKA KIKAPCSOLASA *
10640 REM ************************
10650 REM
10670 POKE V+17,PEEK(V+17) AND 255-6*16
10675 REM GRAFIKA KIKAPCSOLASA
10680 POKE V+22,PEEK(V+22) AND 255-16
10685 REM TOBBSZINU UM. KIKAPCSOLASA
10690 POKE V+24,PEEK(V+24) AND 255-8
10695 REM JELKESZLETTAR. VISSZA $1000-RE
10696 RETURN
10700 REM ********************
10720 REM * PONTOK SZAMITASA *
10740 REM ********************
10750 REM
10760 RA=320*INT(Y/8)+(Y AND 7)
```

```
10770 BA=8*INT(X/8)
10780 MA=2↑(7-(X AND 7))
10790 AD=GT+RA+BA
10800 POKE AD,PEEK(AD) OR MA
10810 RETURN
11100 REM
11110 REM ***********************
11120 REM * ELLIPSZIS RAJZOLASA *
11130 REM ***********************
11140 REM
11150 FOR F2=-1 TO 1 STEP 2:REM JOBB/BAL-JELZO
11160 FOR I=0 TO F2*(XR) STEP F2
11170 Z=YR*SQR(1-I↑2/XR↑2):X=I+X0:REM KOR EGYENLETE
11180 Y=Y0+Z:GOSUB 10760:Y=Y0-Z:GOSUB 10760
11185 REM ALSO/FELSO PONT
11190 NEXT I,F2:RETURN
```

A programban előforduló változók:

XO=YO: az ellipszis középpontjának koordinátái,

XR/YR: az x és y irányú sugár (a és b).

4.3 A sprite-ok programozása

Számítógépünk egyik legkiválóbb tulajdonsága, hogy a képernyőn 8, egymástól független sprite-ot tud megjeleníteni és mozgatni. Aki elolvasta az erre vonatkozó 3.5 fejezetet, annak már van némi elképzelése a sprite-ok szervezésének és megjelenítésének hardverszintű előfeltételeiről. Ebben a részben megtanuljuk ezeket a dolgokat programban alkalmazni.

4.3.1 A sprite-ok létrehozása

Az első probléma, ami a sprite-ok programozása során felmerül, az maga a sprite létrehozása. Ez minden további művelet előfeltétele. Már önmagában is merész vállalkozás, mert a sprite-ok elhelyezése a tárban elég bonyolult dolog.

A 3.5 fejezetben már megbeszéltük, hogy a sprite-ok 24×21 (többszínű üzemmódban 12×21) pontból állnak. A tárban minden pontot 1 bit (többszínű üzemmódban 2 bit) képvisel. Minden 8 pont 1 byte-ot alkot, így 1 sor információit 24/8 = 3 byte-ban tárolhatjuk. Ez a három byte a tárban közvetlenül egymás után helyezkedik el. A következő három byte a második sort jelenti... stb.

A sprite-ok tervezéséhez ún. sprite-adatlapot használunk, aminek mintáját a Függelékben közöljük. Ezt érdemes több példányban is lemásolni.

Ezen az adatlapon bejelöljük a sprite-ot alkotó bekapcsolt pontokat, így végül a jövőbeli sprite-unk tökéletes mását kapjuk.

Jegyezzük meg! A sprite-ot mindig úgy tervezzük, hogy ne legyenek benne egyedülálló pontok. Az egyenes rajzolásánál már említettük, hogy azok a pontok, amelyeknek vízszintes irányban nincs szomszédjuk, csak gyengén, esetleg egyáltalán nem láthatók.

Ez többszínű üzemmódban nem érvényes, hiszen ott úgyis dupla szélességűek a pontok.

Ha az adatlap kitöltésével végeztünk, ki kell számítanunk az ún. bitmintát. Ez a következőképpen lehetséges: a lapon bejelölt bekapcsolt pontok a tár egy-egy bekapcsolt bitjét jelentik. A biteket nyolcasával összefogva kiszámítjuk a sprite-ok meghatározó 63 byte bináris értékeit, amelyeket a Függelékben található konverziós táblázat segítségével decimális számokká alakítunk.

A BASIC több lehetőséget is kínál a sprite-minták tárolására. Ezek közül a legegyszerűbb, amikor a 63 értéket DATA-sorokban rögzítjük és READ-del olvassuk be. Az adatokat helytakarékossági okokból folyamatosan egymás után is írhatjuk, de az áttekinthetőség érdekében célszerűbb az ilyen megoldás:

```
1 REM *** P 19 ***
2 REM
1000 DATA 000,000,000
1010 DATA 000,000,000
1020 DATA 002,000,064
1030 DATA 001,000,128
1040 DATA 000,129,000
1050 DATA 000,066,000
1060 DATA 000,060,000
1070 DATA 000,126,000
1080 DATA 000,195,000
1090 DATA 001,141,128
1100 DATA 003,044,192
1110 DATA 031,255,248
1120 DATA 062,153,124
1130 DATA 125,066,190
1140 DATA 255,255,255
1150 DATA 001,255,128
1160 DATA 001,189,128
1170 DATA 003,060,192
1180 DATA 015,000,240
1190 DATA 015,000,240
1200 DATA 000,000,000
```

Ebből persze nem derül ki rögtön, hogy egy járműről van szó, de a 3×21-es byte-szerkezet világosan látható.

Ebben az elrendezésben mindegyik DATA-sor a sprite egy sorát jelenti. Az adatok beolvasásához írjuk a DATA-sorok elé a következő néhány utasíssort:

```
1 REM *** P 20 ***
2 REM
100 AD=13*64:REM 13-AS BLOKK CIME
110 FOR X=0 TO 62
120 READ DT:REM A 63 ADAT BEOLVASASA
130 POKE AD+X,DT:REM BEIRAS A 13-AS BLOKKBA
140 NEXT X
```

Ez a kis program sorban beolvassa a 63 adatot és beírja a tárba a megadott címtől (AD) kezdődően.

(Itt jegyezzük meg, hogy a könyvben közölt programok 1-es és 2-es sora nem tartozik szorosan a programokhoz, csupán a könyvben való tájékozódás a célja. A programok összefűzésekor ezeket természetesen el kell hagyni!)

Ez a módszer meglehetősen nagy tárkapacitást igényel. Sokkal jobb megoldás, ha a sprite-okat mágneslemezen vagy kazettán rögzítjük, pl. szekvenciális file-ként. Ebben az esetben azt a program bármely részén a perifériáról olvashatjuk be. Ezzel a módszerrel a lemezen vagy kazettán egész adatbankot hozhatunk létre, amelyből a program mindig csak a szükséges részeket tölti be.

A 63 adatot ilyenkor is DATA-sorokban tároljuk, amelyeket a következő programmal, szekvenciális file-ként a lemezre rögzíthetünk:

```
1 REM *** P 21 ***
2 REM
10 OPEN 1,8,2,"SPRITE,S,W"
15 REM SEQ. FILE MEGNYITASA IRASRA
20 FOR X=0 TO 62
30 READ DT:REM A 63 ADAT BEOLVASASA
40 PRINT#1,CHR$(DT):REM IRAS A LEMEZRE
50 NEXT X           :REM (ASCII FORMABAN)
60 CLOSE 1:REM FILE LEZARASA
```

A file neve: SPRITE.

A betöltés a következő programmal lehetséges:

```
1 REM *** P 22 ***
2 REM
10 AD=13*64:REM A 13-AS BLOKK CIME
20 OPEN 1,8,2,"SPRITE,S,R"
25 REM SEQ. FILE MEGNYITASA OLVASASRA
30 FOR X=0 TO 62
40 INPUT#1,DT$:REM ADATOK BEOLVASASA (ASCII)
50 POKE AD+X,ASC(DT$+CHR$(0)):REM ES BEIRASA
60 NEXT X
70 CLOSE 1:REM FILE LEZARASA
```

A sprite-okat ezenkívül programfile-ként is rögzíthetjük. Tagadhatatlan, hogy ez az egész elég komplikált, és akiknek nincs elég gyakorlatuk, azoknak rengeteg fejtörést okoz. Ebből kiindulva készítettünk egy programot, amivel megkönnyíthetjük a sprite-ok előállítását. Ez tulajdonképpen egy sprite-editor, amit BASIC-ben is és gépi nyelven is közreadunk.

A rögzített sprite-ot a P22-es program szerint olvashatjuk vissza, ha a 20-as sort a következőképpen módosítjuk:

20 OPEN 1,8,2, "SPRITE, P,R" : REM PROGRAM-FILE MEGNYITASA OLVASASRA

Kétségtelen, hogy a program begépelése rengeteg munkát jelent, de ez bőven megtérül a használatkor.

Mielőtt először elindítanánk, tároljuk lemezre, mert a programfutás közben módosulni fog néhány BASIC-mutató. Ha a DATA-sorok bevitelekor hibát vétettünk, a számítógép hibaüzenettel figyelmeztet.

```
1 REM *** P 23 ***
2 REM
100 REM ******************
110 REM * SPRITE TERVEZO *
120 REM *                *
130 REM *  SEGEDPROGRAM  *
140 REM ******************
150 REM
160 REM ELOKESZULETEK
170 REM *************
180 GOSUB 2730:REM GEPI ALPROGRAM BEOLVASASA
190 POKE 53280,0:POKE 53281,0
195 REM HATTER ES KERET FEKETE
200 POKE 650,255:REM MINDEN JEL ISMETLODIK
210 POKE 45,0:POKE 46,80:RUN 220
215 REM BASIC TAROLO VEGE=$5000
220 REM
```

```
230 REM GEPI KODU ALPROGRAMOK
240 REM *********************
250 IN%=18432 :REM ELOKESZITES
260 PU%=18632 :REM PONT BEJELOLESE
270 NE%=18567 :REM KOORDINATARENDSZER
280 LA%=18503 :REM JELKESZLET TOLTESE
290 SP%=18531 :REM JELKESZLET MENTESE
300 CA%=18758 :REM TARTALOMJEGYZEK
310 BE%=18712 :REM UTASITAS AZONOSITASA
320 IV%=18830 :REM INVERTALAS
330 VR%=18844 :REM ELTOLAS JOBBRA
340 VL%=18870 :REM ELTOLAS BALRA
350 VO%=18895 :REM ELTOLAS FEL
360 VU%=18935 :REM ELTOLAS LE
370 Q  =  704 :REM SPRITE-BLOKK CIME
380 V  =53248 :REM VIC KEZDOCIME
390 REM
400 REM VEZERLOJELEK
410 REM ************
420 C0$=CHR$(147) :REM KEPERNYOTORLES
430 C1$=CHR$( 19) :REM HOME
440 C2$=CHR$(183) :REM FELSO VONAL
450 C8$=CHR$( 99)+CHR$( 99)+CHR$( 99)
455 C3$=CHR$(117)+C8$+CHR$(105):REM FELSO SZEL 1.
460 C4$=CHR$(106)+C8$+CHR$(107):REM ALSO  SZEL 1.
470 C5$=CHR$(117)+C8$+C8$+CHR$(105):REM F.SZEL 2.
480 C8$=CHR$(106)+C8$+C8$+CHR$(107):REM A.SZEL 2.
490 C9$=CHR$( 98):REM KOZEPSOFUGGOLEGES VONAL
500 C6$=CHR$( 18):REM RVS ON
510 C7$=CHR$(146):REM RVS OFF
520 NA%=828 :REM FILE-NEV HOSSZA ($C800)
530 GA%=186 :REM KESZULEKCIM ($BA)
540 TA%=821 :REM BILLENTYU/UTASITASKOD
550 SG%=  1 :REM A SPRITE NAGYSAGA
560 YK%=822 :REM Y-KOORD.
570 XK%=823 :REM X-KOORD.
580 REM
590 REM SZINEK DEFINIALASA
600 REM ******************
610 DATA 144,  5, 28,159,156, 30, 31,158
620 DATA 129,149,150,151,152,153,154,155
630 DIM C$(16)
635 FOR Y=0 TO 15:READ X:C$(Y)=CHR$(X):NEXT Y
640 N=1:F(0)=0:F(1)=1:V$=" ":SYS IN%:REM SZIN/INIC
650 REM
660 REM TORLES
670 REM ******
680 SYS IN% :REM SPRITE TORLESE
690 PRINT C0$
700 PRINT C1$;SPC(13);C$(7);"SPRITE-TERVEZES"
710 PRINT SPC(12);C$(1);" (C) AXEL PLENGE"
720 PRINT C$(4);:FOR X=1 TO 40:PRINT C2$;:NEXT X
730 PRINT C$(7)"  7"C$(6)"6543210"C$(7)"7"C$(6);
735 PRINT"6543210"C$(7)"7"C$(6)"6543210";
740 SYS NE%:REM HALO RAJZOLASA
```

```
750 GOSUB 1820:PRINT:PRINT:REM ALLAPOTMEZO
760 PRINT:PRINT:PRINT TAB(30);C3$
770 FOR X=1 TO 3:PRINT TAB(30);C9$;"   "C9$:NEXT X
775 PRINT TAB(30);C4$:REM 1-ES MINTA
780 PRINT TAB(27);"2";C5$;"2"
785 FOR X=1 TO 5:PRINT TAB(27);" ";C9$;"       ";
786 PRINT C9$:NEXT X
790 PRINT TAB(27);" ";C8$;"  ";:REM 2-ES MINTA
800 POKE 53248+21,3:X=0:Y=0:REM SPRITE BE;X=0,Y=0
810 REM
820 REM BEVITELI CIKLUS
830 REM ***************
840 A=X+2:B=Y+4:GOSUB 2450:REM POZICIONALAS
850 POKE XK%,X:POKE YK%,Y:F=0:REM KOORD.-K ATADASA
860 PRINT C$(7);C6$;" ";CHR$(157);:REM VILLOGAS BE
870 FOR S=1 TO 50:GET A$:IF A$<>"" THEN 890
880 NEXT S:SYS PU%
885 FOR S=1 TO 50:GETA$:IFA$="" THEN NEXTS:GOTO860
886 REM KIKAPCSOLAS
890 REM
900 REM UTASITAS AZONOSITAS
910 REM *******************
920 SYS PU%:C=ASC(A$):POKE TA%,C:SYS BE%
925 S=PEEK(TA%):REM UTASITAS ATADAS/VISSZAJELZES
930 REM RESZLETEZES:
940 ON S GOTO 1050,1050,1070,1070,1090,1090
950 ON S- 6 GOTO 1110,1110,1910,1910,1910,1910
960 ON S-12 GOTO 1910,1910,1910,1910, 650,1360
970 ON S-18 GOTO 1450,1490,1570,1130,2150
980 ON S-23 GOTO 1200,1970,1240, 810
990 REM
1000 REM UTASITAS FELDOLGOZAS
1010 REM ********************
1020 REM
1030 REM KURZOR MOZGATAS
1040 REM ***************
1050 X=X+1:IF X=24 THEN X=0:GOTO 1090
1060 GOTO 810:REM JOBBRA
1070 X=X-1:IF X<0 THEN X=23:GOTO 1110
1080 GOTO 810:REM BALRA
1090 Y=Y+1:IF Y=21 THEN Y=0
1100 GOTO 810:REM LE
1110 Y=Y-1:IF Y<0 THEN Y=20
1120 GOTO 810:REM FEL
1130 REM
1140 REM BEFEJEZES
1150 REM *********
1160 A=2:B=15:GOSUB 2450:REM POZICIONALAS
1170 PRINT C6$;C$(7);"VEGE (I/N)?";C7$;C$(6)
1180 INPUT T$:IF T$="I" THEN SYS 64738
1190 GOTO 690
1200 REM
1210 REM TARTALOMJEGYZEK
1220 REM ***************
1230 PRINT C0$:SYS CA%:GOSUB 2490:GOTO 690
```

```
1240 REM
1250 REM ELTOLAS
1260 REM *******
1270 GOSUB 2530:GOSUB 2440:PRINT C$(1);"ELTOLAS :"
1280 PRINT TAB(27)"     ":PRINT TAB(27)"JOBBRA (J)"
1285 PRINT TAB(27)"BALRA  (B)"
1290 PRINT TAB(27)"FEL    (F)"
1295 PRINT TAB(27)"LE     (L)"
1300 GOSUB 2490
1310 IF T$="J" THEN SYS VR%:GOTO 1350
1320 IF T$="B" THEN SYS VL%:GOTO 1350
1330 IF T$="F" THEN SYS VO%:GOTO 1350
1340 IF T$="L" THEN SYS VU%
1350 GOSUB 2530:GOTO 700
1360 REM
1370 REM A SPRITE NAGYSAGA
1380 REM *****************
1390 ON SG%+1 GOSUB 1410,1420,1430,1440
1400 POKE V+23,A:POKE V+29,B:SG%=(SG%+1) AND 3
1405 GOTO 810
1410 A=2:B=2:RETURN
1420 A=0:B=2:RETURN
1430 A=2:B=0:RETURN
1440 A=0:B=0:RETURN
1450 REM
1460 REM INVERTALAS
1470 REM **********
1480 SYS IV%:GOTO 700
1490 REM
1500 REM MENTES
1510 REM ******
1520 GOSUB 2530
1530 GOSUB 2440:PRINT C6$;C$(1);"SPRITE    "
1535 PRINT TAB(27);C6$;"MENTESE     ";C7$
1540 GOSUB 1700:IF F=1 THEN F=0:GOTO 1490
1545 REM BEVITEL/HIBAELLENORZES
1550 IF F=2 THEN F=0: GOTO 1630 REM HIBA
1560 SYS SP%:GOTO 1650:REM MENTES
1570 REM
1580 REM BETOLTES
1590 REM ********
1600 GOSUB 2530
1610 GOSUB 2440:PRINT C6$;C$(1);"SPRITE"
1615 PRINT TAB(27);C6$;"TOLTESE";C7$
1620 GOSUB 1700:IF F=1 THEN F=0:GOTO 1570
1630 IF F=2 THEN F=0:GOTO 690
1640 SYS LA%
1650 REM HIBAELLENORZES (CSAK LEMEZRE!)
1660 OPEN 1,8,15:INPUT#1,DS,DS$,DT,DB:CLOSE1
1670 IF DS<20 THEN 690:REM OK
1680 PRINT
1685 T$=STR$(DS)+","+DS$+","+STR$(DT)+","+STR$(DB)
1690 GOSUB 2600:PRINT T$:FOR S=1 TO 2000:NEXT S
1695 GOTO 690:REM VILLOGAS
1700 REM
```

```
1710 REM FILE-NEV MEGEDASA
1720 REM ****************
1730 A$="":PRINT:PRINT TAB(27)"FILE-NEV"C$(6)
1735 PRINT TAB(27);:INPUT A$:T=LEN(A$)
1740 S=VAL(RIGHT$(A$,1))
1750 IFS<>0ANDLEFT$(RIGHT$(A$,2),1)=","THENT=T-2:POKEGA,S
1760 IF T=0 THEN F=2:RETURN:REM NINCS NEV
1770 IF T>17 THEN 1800
1780 REM NEV A GEPI RUTINNAK
1790 POKE NA%,T
1792 FOR S=1 TO T
1796 POKE NA%+S,ASC(MID$(A$,S,1)):NEXT S
1798 RETURN
1800 PRINT CHR$(145);:GOSUB 6535
1805 T$=C6$+"HOSSZU!"+C7$:GOSUB 2590:REM HIBAUZENET
1810 PRINT C$(6):F=1:RETURN
1820 REM
1830 REM ALLAPOTMEZO ELOALLITASA
1840 REM ***********************
1850 A=27:B=4:GOSUB 2450
1860 GOSUB 2530
1870 A=27:B=5:GOSUB 2450
1880 PRINT TAB(27);C$(7);"SZINEK:"
1890 PRINT TAB(27);C$(2);:FOR S=1 TO 7
1895 PRINT CHR$(163);:NEXT S:PRINT C$(6)
1900 FOR S=0 TO 1:PRINT TAB(27);"ASZ. ";S;":";F(S)
1905 NEXT S:RETURN
1910 REM
1920 REM RAJZOLAS
1930 REM ********
1940 S=((S-12) AND 2)/2:REM ???
1950 T=X/8:AD=INT(T):T=2↑(7-8*(T-AD)):AD=Y*3+AD+Q
1960 POKE AD,PEEK(AD) AND(255-T) OR S*T:GOTO 810
1970 REM
1980 REM SZINSKALA
1990 REM *********
2000 PRINT CHR$(147)
2010 A=0:B=4:GOSUB 2450
2012 PRINT TAB(4);C$(1)"S"C$(2)"Z"C$(3)"I";
2014 PRINT C$(4)"N"C$(5)"S";
2020 PRINT C$(6)"K"C$(7)"A"C$(4)"L"C$(6)"A";
2025 PRINT C$(2)" "C$(7)":"
2030 PRINT TAB(4);C$(1);CHR$(172);
2032 FOR S=1 TO 32:PRINT CHR$(162);:NEXT S
2034 PRINT CHR$(187)
2040 FOR S=1 TO 2:PRINT TAB(4);C6$;CHR$(161);
2050 FOR T=0 TO 15:PRINT C$(T);"  ";:NEXT T
2055 PRINT C$(1);C7$;CHR$(161):NEXT S
2060 PRINT TAB(4);C6$;CHR$(161);
2062 PRINT" 0 1 2 3 4 5 6 7 8 9101112131415";
2064 PRINT C7$;CHR$(161)
2070 PRINT:PRINT C$(6);"    ALAPSZINEK (F1/F3): ";
2080 GOSUB 2490:T=ASC(T$)-133:REM FUNKCIOBILL.-K
2090 IF  T<0 OR T>1 THEN GOSUB 2590:GOTO 690
2100 IF T>1 THENT=T-4
```

```
2110 PRINT T:T$="":INPUT"     SZIN ";T$
2115 S=ABS(INT(VAL(T$)))
2120 IF T$="" OR S>15 THEN GOSUB 2590:GOTO 690
2130 F(T)=S:POKE V+3,F(0):REM HATTERSZINBEALLITAS
2140 POKE V+39,F(1):POKE V+40,F(1):GOTO 690
2145 REM SPRITE-SZINEK BEALLITASA
2150 REM
2160 REM UTASITASKESZLET
2170 REM ***************
2180 POKEV+21,0:REM SPRITE KI
2190 PRINT C0$;C6$;C$(2)"                        ";C$(7);
2200 PRINT"UTASITASKESZLET";C$(2);"                    ";
2205 PRINT C7$;
2210 PRINT C$(4);:FOR S=1 TO 40:PRINT CHR$(184);
2215 NEXT S:PRINT
2220 PRINT C$(1)"SZAM  "C6$"UTASITAS"C7$" -";
2225 PRINT C$(5)"                   FELADATA"C$(4)
2230 FOR S=1 TO 10:PRINT"----";:NEXT S
2240 PRINT C$(1)" (1)  "C6$"()<...) "C7$" -";
2245 PRINT C$(5)" A KURZOR MOZGATASA    "
2250 PRINT C$(1)"        "C6$"(2QWA ) "C7$;C$(5)
2260 PRINT C$(1)" (2)  "C6$"(F1-F8) "C7$" -";
2265 PRINT C$(5)" RAJZ A 0-15 SZINEKKEL"
2270 PRINT C$(1)" (3)  "C6$"(F)     "C7$" -";
2275 PRINT C$(5)" F1/F3 SZINEK (0-15)   "
2280 PRINT C$(1)" (4)  "C6$"(B)     "C7$" -";
2285 PRINT C$(5)" UTASITASKESZLET       "
2290 PRINT C$(1)" (5)  "C6$"(G)     "C7$" -";
2295 PRINT C$(5)" A SPRITE NAGYITASA    "
2300 PRINT C$(1)" (6)  "C6$"(I)     "C7$" -";
2305 PRINT C$(5)" A SPRITE INVERTALASA "
2310 PRINT C$(1)" (7)  "C6$"(V)     "C7$" -";
2315 PRINT C$(5)" A SPRITE ELTOLASA     "
2320 PRINT C$(1)" (8)  "C6$"(L)     "C7$" -";
2325 PRINT C$(5)" A SPRITE TORLESE      "
2330 PRINT C$(1)" (9)  "C6$"(CTRL/G)"C7$" -";
2335 PRINT C$(5)" A SPRITE BETOLTESE    "
2340 PRINT C$(1)"(10)  "C6$"(CTRL/S)"C7$" -";
2345 PRINT C$(5)" A SPRITE MENTESE      "
2350 PRINT C$(1)"(11)  "C6$"(C)     "C7$" -";
2355 PRINT C$(5)" TARTALOMJEGYZEK       "
2360 PRINT C$(1)"(12)  "C6$"(CTRL/X)"C7$" -";
2365 PRINT C$(5)" BEFEJEZES             "
2370 GOSUB 2490:POKE V+21,3:GOTO 690:REM SPRITE BE
2380 REM
2390 REM ALPROGRAMOK
2400 REM ***********
2410 REM
2420 REM POZICIONALAS
2430 REM ************
2440 A=27:B=5:REM UZENET-MEZO
2450 PRINT C1$;:FOR S=2 TO B:PRINT:NEXT S
2455 PRINT TAB(A);:RETURN
2460 REM
2470 REM BEVITEL A BILLENTYUZETROL
```

```
2480 REM ************************
2490 POKE198,0:WAIT 198,255:GET T$:RETURN
2500 REM
2510 REM UZENET-MEZO TORLESE
2520 REM ******************
2530 GOSUB 2440
2540 FOR S=1 TO 6:PRINT TAB(27);
2542 FOR T=1 TO 4:PRINT"    ";:NEXT T:PRINT
2544 NEXT S:REM TORLES
2550 RETURN
2560 REM
2570 REM HIBAUZENET (VILLOGAS)
2580 REM *********************
2590 T$="NEM MEGENGEDETT !"
2600 A=4:B=18:GOSUB 2450:PRINT C$(1)
2602 FOR S=1 TO 9:PRINT TAB(4)T$:GOSUB 2630
2604 PRINT CHR$(145);
2610 PRINT TAB(4)"                    ";
2620 PRINT CHR$(145):GOSUB 2630:NEXT S
2625 PRINT TAB(4)"                    ":F=1:RETURN
2630 FOR T=1 TO 75:NEXT T:RETURN:REM VARAKOZAS
2640 REM
2650 REM **********************
2660 REM **                  **
2670 REM ** GEPI KODU RUTINOK **
2680 REM **                  **
2690 REM **********************
2700 REM
2710 REM INDITASKOR AZ ADATOK TORLODNEK !!!
2720 REM
2730 FOR I=1 TO 16:READ X:NEXT I
2735 REM AZ ELSO 16 ADAT ATUGRASA (SZINEK)
2740 FOR I=18432 TO 18969
2750 READ X:POKE I,X:S=S+X:NEXT I
2760 DATA 162, 62,169,  0,157,192
2765 DATA   2,202, 16,250,169, 11
2770 DATA 141,248,  7,141,249,  7
2775 DATA 169, 16,141,  0,208,169
2780 DATA 163,141,  1,208,169,  7
2785 DATA 141,  2,208,169,201,141
2790 DATA   3,208,169,  3,141, 16
2795 DATA 208,141, 21,208,169,  2
2800 DATA 141, 23,208,141, 29,208
2805 DATA 169,  0,141, 27,208,141
2810 DATA  28,208,169,  1,141, 39
2815 DATA 208,141, 40,208, 96,173
2820 DATA  60,  3,162, 61,160,  3
2825 DATA  32,249,253,169,  2,166
2830 DATA 186,160,  0, 32,  0,254
2835 DATA 169,  0,162,192,160,  2
2840 DATA  76,213,255,173, 60,  3
2845 DATA 162, 61,160,  3, 32,249
2850 DATA 253,169,  2,166,186,160
2855 DATA   0, 32,  0,254,169,192
2860 DATA 133,  2,169,  2,133,  3
```

```
2865 DATA 169,  2,162,255,160,  2
2870 DATA  76,216,255,160,  0,169
2875 DATA  13, 32,210,255,152, 72
2880 DATA  56,233, 10,144, 12,168
2885 DATA 233, 10,144,  4,168,169
2890 DATA  50, 44,169, 49, 44,169
2895 DATA  32, 32,210,255,152,  9
2900 DATA  48, 32,210,255,104,168
2905 DATA 162,  0, 32,206, 72,169
2910 DATA  29, 32,210,255,232,224
2915 DATA  24,208,243,169,165, 32
2920 DATA 210,255,200,192, 21,208
2925 DATA 194, 96,174, 55,  3,172
2930 DATA  54,  3,138, 72,152, 72
2935 DATA 133,  2, 10,101,  2,133
2940 DATA   2,138,160,255, 56,233
2945 DATA   8,200,176,251,105,  8
2950 DATA 170,152, 24,101,  2,168
2955 DATA 169,  0, 56,106,202, 16
2960 DATA 252, 57,192,  2,208,  3
2965 DATA 160,  0, 44,160,  6,162
2970 DATA   6,185, 12, 73, 32,210
2975 DATA 255,200,202,208,246,104
2980 DATA 168,104,170, 96,146, 31
2985 DATA 111, 31,146,157, 18, 28
2990 DATA  32, 31,146,157,173, 53
2995 DATA   3,162,  0,160, 28,232
3000 DATA 221, 43, 73,240,  3,136
3005 DATA 208,247,142, 53,  3, 96
3010 DATA  87, 29, 81,157, 65, 17
3015 DATA  50,145,133,137,134,138
3020 DATA 135,139,136,140, 76, 71
3025 DATA  73, 19,  7, 24, 66, 67
3030 DATA  70, 86,169, 36,133,  2
3035 DATA 169,  1,162,  2,160,  0
3040 DATA  32,249,253,169,  2,166
3045 DATA 186,160,  0, 32,  0,254
3050 DATA 169,  0,162,  0,160, 64
3055 DATA 134, 95,132, 96, 32,213
3060 DATA 255,165, 95,164, 96, 32
3065 DATA  55,165,173,  0,  3, 72
3070 DATA 173,  1,  3, 72,169, 61
3075 DATA 141,  0,  3,169,227,141
3080 DATA   1,  3, 32,195,166,104
3085 DATA 141,  1,  3,104,141,  0
3090 DATA   3, 96,162, 62,189,192
3095 DATA   2, 73,255,157,192,  2
3100 DATA 202, 16,245, 96,162,  0
3105 DATA 160,  3, 24,126,192,  2
3110 DATA 232,136,208,249,169,  0
3115 DATA 106, 29,189,  2,157,189
3120 DATA   2,224, 63,208,233, 96
3125 DATA 162, 63,160,  3, 24, 62
3130 DATA 191,  2,202,136,208,249
3135 DATA 169,  0, 42, 29,194,  2
```

```
3140 DATA 157,194,  2,138,208,234
3145 DATA  96,162, 62,134,  2,160
3150 DATA  21,132,  3,166,  2,189
3155 DATA 132,  2, 72,189,192,  2
3160 DATA 168,104,157,192,  2,152
3165 DATA 202,202,202,198,  3,208
3170 DATA 239,166,  2,202,134,  2
3175 DATA 224, 59,208,221, 96,162
3180 DATA   2,134,  2,160, 21,132
3185 DATA   3,166,  2,189,252,  2
3190 DATA  72,189,192,  2,168,104
3195 DATA 157,192,  2,152,232,232
3200 DATA 232,198,  3,208,239,198
3205 DATA   2, 16,226, 96
3210 IF S=61707 THEN PRINT "RENDBEN !":RETURN
3220 PRINT"! HIBA A DATA-SOROKBAN !":END
```

A következőkben közreadjuk ugyanennek programnak a gépi nyelvű megfelelőjét:

```
                          ;
                          ;*** P 24 ***
                          ;
                          ;
                          ;
                          ;GEPI RUTIN
                          ;**********
                          ;
                          ;
                          ;
                          ;
90:     4800                      *=     $4800    ;KEZDOCIM
                          ;
                          ;UGRASI CIMEK ES REGISZTEREK
                          ;***************************
                          ;
140:    4800             CONASS   =     1
150:    4800             CHROUT   =     $FFD2    ;JEL KIVITEL
160:    4800             FNPAR    =     $FDF9    ;FILENEV PARAMETEREK
170:    4800             FPAR     =     $FE00    ;FILEPARAMETEREK
180:    4800             SAVE     =     $FFD8    ;MENTES LEMEZ/KAZETTA
190:    4800             LOAD     =     $FFD5    ;BETOLTES LEMEZ/KAZETTA
200:    4800             V        =     $D000    ;VIDEOVEZERLO
210:    4800             BLOKK    =     704      ;11-ES SPRITE BLOKK
220:    4800             BLOKKSZ  =     11       ;BLOKKSZAM 11
230:    4800             HOSSZ    =     $033D    ;FILENEV HOSSZA
240:    4800             UEMM     =     $0334    ;TAROLO UZEMMOD
250:    4800             BILL     =     $0335    ;UTASITAS BILLENTYUK
260:    4800             YKOORD   =     $0336    ;MEZO-Y-KOORDINATA
270:    4800             XKOORD   =     $0337    ;MEZO-X-KOORDINATA
                          ;
                          ;MINTASPRITE TORLESE+PARAMETEREK
                          ;*******************************
```

```
320:    4800 A2 3E     INIC    LDX  #62       ;63 BYTE
330:    4802 A9 00             LDA  #$00
340:    4804 9D C0 02 I1       STA  BLOKK,X   ;11-ES BLOKK TORLESE
350:    4807 CA                DEX
360:    4808 10 FA             BPL  I1
370:    480A A9 0B             LDA  #BLOKKSZ  ;11-ES BLOKK
380:    480C 8D F8 07          STA  $07F8     ;SPRITEMUTATO 0-ROL 11-RE
390:    480F 8D F9 07          STA  $07F9     ;SPRITEMUTATO 1-ROL 11-RE
400:    4812 A9 10             LDA  #16
410:    4814 8D 00 D0          STA  V+0       ;0-AS SP X-KOORD FELSO BY
420:    4817 A9 A3             LDA  #163
430:    4819 8D 01 D0          STA  V+1       ;Y KOORDINATA
440:    481C A9 07             LDA  #7
450:    481E 8D 02 D0          STA  V+2       ;1-ES SP X-KOORD FELSO BY
460:    4821 A9 C9             LDA  #201
470:    4823 8D 03 D0          STA  V+3       ;Y KOORDINATA
480:    4826 A9 03             LDA  #%00000011  ;X-K FELSO BYTE=1
490:    4828 8D 10 D0          STA  V+16
500:    482B 8D 15 D0          STA  V+21      ;SPRITE BEKAPCSOLASA
510:    482E A9 02             LDA  #%00000010
520:    4830 8D 17 D0          STA  V+23      ;1-ES SP NAGYITASA(Y)
530:    4833 8D 1D D0          STA  V+29      ;1-ES SP NAGYITASA(X)
540:    4836 A9 00             LDA  #$00
550:    4838 8D 1B D0          STA  V+27      ;SPRITE ELSOBBSEG
560:    483B 8D 1C D0          STA  V+28      ;SPRITE NORMAL SZINBEN
570:    483E A9 01             LDA  #$01
580:    4840 8D 27 D0          STA  V+39      ;0-AS SPRITE FEHER
590:    4843 8D 28 D0          STA  V+40      ;1-ES SPRITE FEHER
600:    4846 60                RTS            ;VISSZA

                       ;SPRITE BETOLTESE
                       ;****************
                       ;
650:    4847 AD 3C 03 TOLT     LDA  HOSSZ-1   ;NEV HOSSZA
660:    484A A2 3D             LDX  #<HOSSZ
670:    484C A0 03             LDY  #>HOSSZ   ;FELSO BYTE
680:    484E 20 F9 FD          JSR  FNPAR
690:    4851 A9 02             LDA  #$02      ;LOGIKAI FILESZAM
700:    4853 A6 BA             LDX  $BA       ;KESZULEKCIM
710:    4855 A0 00             LDY  #$00      ;SZEKUNDERCIM
720:    4857 20 00 FE          JSR  FPAR
730:    485A A9 00             LDA  #$00      ;LOAD/VERIFY-JELZO
740:    485C A2 C0             LDX  #<BLOKK   ;KEZDOCIM(ALSO BYTE)
750:    485E A0 02             LDY  #>BLOKK   ;KEZDOCIM(FELSO BYTE)
760:    4860 4C D5 FF          JMP  LOAD
                       ;
                       ;SPRITE TAROLASA
                       ;***************
                       ;
810:    4863 AD 3C 03 MENT     LDA  HOSSZ-1   ;FILENEV HOSSZA
820:    4866 A2 3D             LDX  #<HOSSZ   ;FILENEV-CIM(ALSO)
830:    4868 A0 03             LDY  #>HOSSZ   ;FILENEV-CIM(FELSO)
840:    486A 20 F9 FD          JSR  FNPAR
850:    486D A9 02             LDA  #$02      ;LOGIKAI FILESZAM
860:    486F A6 BA             LDX  $BA       ;KESZULEKCIM
```

```
870:    4871 A0 00             LDY   #$00      ;SZEKUNDERCIM
880:    4873 20 00 FE          JSR   FFAR
960:    4876 A9 C0             LDA   #<BLOKK
970:    4878 85 02             STA   $02       ;KEZDOCIM(ALSO BYTE)
980:    487A A9 02             LDA   #>BLOKK
990:    487C 85 03             STA   $03       ;KEZDOCIM(FELSO BYTE)
1000:   487E A9 02             LDA   #$02      ;A KEZDOCIM NULLASLAP CIMEI
1010:   4880 A2 FF             LDX   #<BLOKK+$3F  ;VEGCIM(ALSO BYTE)
1020:   4882 A0 02             LDY   #>BLOKK+$3F  ;VEGCIM(FELSO BYTE)
1030:   4884 4C D8 FF          JMP   SAVE      ;SPRITE TAROLASA
                         ;
                         ; MUNKAMEZO FELEPITESE
                         ; ******************
                         ;
1500:   4887 A0 00    HALO     LDY   #$00      ;SORSZAMLALO=0
1510:   4889 A9 0D    N0       LDA   #$0D      ;CARRIGE RETURN
1520:   488B 20 D2 FF          JSR   CHROUT
1530:   488E 98                TYA
1540:   488F 48                PHA
1550:   4890 38                SEC
1560:   4891 E9 0A             SBC   #10
1570:   4893 90 0C             BCC   N1        ;SZOKOZ Y<10-NEL
1580:   4895 A8                TAY
1590:   4896 E9 0A             SBC   #10
1600:   4898 90 04             BCC   N2        ;Y<20
1610:   489A A8                TAY
1620:   489B A9 32             LDA   #$32
1630:   489D 2C                .BYTE$2C        ;KOVETKEZO UTASITAS ATUGRASA
1640:   489E A9 31    N2       LDA   #$31
1650:   48A0 2C                .BYTE$2C        ;KOVETKEZO UTASITAS ATUGRASA
1660:   48A1 A9 20    N1       LDA   #$20      ;" " SZOKOZ
1670:   48A3 20 D2 FF          JSR   CHROUT    ;KIVITEL
1680:   48A6 98                TYA             ;MARADEK
1690:   48A7 09 30             ORA   #$30      ;ASCII ELOALLITASA
1700:   48A9 20 D2 FF          JSR   CHROUT    ;KIVITEL
1710:   48AC 68                PLA
1720:   48AD A8                TAY
1730:   48AE A2 00             LDX   #$00
1740:   48B0 20 CE 48 N3       JSR   PONT      ;EGY PONT RAJZOLASA
1750:   48B3 A9 1D             LDA   #$1D      ;KURZOR JOBBRA
1760:   48B5 20 D2 FF          JSR   CHROUT
1770:   48B8 E8                INX             ;KOVETKEZO PONT
1780:   48B9 E0 18             CPX   #24       ;24 PONT/SOR
1790:   48BB D0 F3             BNE   N3
1800:   48BD A9 A5             LDA   #$A5      ;VONAL (CHR$(165))
1810:   48BF 20 D2 FF          JSR   CHROUT
1820:   48C2 C8                INY             ;KOVETKEZO SOR
1830:   48C3 C0 15             CPY   #21       ;21 SOR
1840:   48C5 D0 C2             BNE   N0
1850:   48C7 60                RTS
                         ;
                         ;EGY KOORDINATA RAJZOLASA
                         ;************************
                         ;
```

```
48C8 AE 37 03 PONT2     LDX   XKOORD   ;VEZERLES BASICBOL
48CB AC 36 03           LDY   YKOORD   ;X/Y-KOORDINATAK
48CE 8A       PONT      TXA
48CF 48                 PHA
48D0 98                 TYA
48D1 48                 PHA            ;KOORDINATAK MENTESE
48D2 85 02              STA   $02
48D4 0A                 ASL   A
48D5 65 02              ADC   $02      ;Y-KOORDINATA*3
48D7 85 02              STA   $02      ;ES MENTES
48D9 8A                 TXA
48DA A0 FF              LDY   #$FF     ;SZAMLALO=0
48DC 38                 SEC
48DD E9 08    P3        SBC   #$08
48DF C8                 INY
48E0 B0 FB              BCS   P3       ;X-KOORDINATA/8
48E2 69 08              ADC   #$08     ;MARADEK
48E4 AA                 TAX
48E5 98                 TYA
48E6 18                 CLC
48E7 65 02              ADC   $02      ;Y*3+INT(X/8)
48E9 A8                 TAY
48EA A9 00              LDA   #$00
48EC 38                 SEC            ;EGY BIT BEKAPCSOLASA
48ED 6A       P0        ROR   A        ;VALODI BIT KIKERESESE
48EE CA                 DEX            ;MARADEK CSOKKENTESE
48EF 10 FC              BPL   P0
48F1 39 C0 02           AND   BLOKK,Y ;SPRITEBYTE TOBBI BIT TORLESE
48F4 D0 03              BNE   P1       ;BIT(=PONT) BE
48F6 A0 00              LDY   #$00
48F8 2C                 .BYTE$2C       ;BIT UTASITAS
48F9 A0 06    P1        LDY   #$06     ;BIT BE
48FB A2 06              LDX   #$06
48FD B9 0C 49 P2        LDA   PKTTAB,Y ;JEL A TABLAZATBOL
4900 20 D2 FF           JSR   CHROUT
4903 C8                 INY
4904 CA                 DEX
4905 D0 F6              BNE   P2       ;KOVETKEZO JEL
4907 68                 PLA
4908 A8                 TAY
4909 68                 PLA
490A AA                 TAX            ;KOORDINATAK ISMETLESE
490B 60                 RTS
                        ;
                        ;JELEK EGY KOORDINATAHOZ
                        ;***********************
                        ;
490C 92 1F 6F PKTTAB    .BYTE146,031,111,031,146,157
4912 12 1C 20           .BYTE018,028,032,031,146,157
                        ;
                        ;UTASITASBILLENTYU AZONOSITASA
                        ;*****************************
                        ;
4918 AD 35 03 BEFIDE    LDA   BILL     ;BILLENTYUKOD
491B A2 00              LDX   #$00     ;UTASITASKOD
```

```
2450:   491D A0 1C             LDY   #$1C     ;27-1 UTASITAS(SZAMLALO)
2460:   491F E8       B1       INX            ;UTASITASKOD NOVELESE
2470:   4920 DD 2B 49          CMP   UTTAB-1,X ;BILL OSSZEHAS A TABL-TAL
2480:   4923 F0 03             BEQ   B2       ;MEGTALALTA
2490:   4925 88                DEY            ;KOVETKEZO UTASITAS
2500:   4926 D0 F7             BNE   B1       ;MEG NINCS KESZ
2510:   4928 8E 35 03 B2       STX   BILL     ;UTASITASKOD VISSZAJELZES
2520:   492B 60                RTS            ;VISSZA A BASICBE
                      ;
                      ;UTASITASBILLENTYU TABLAZAT
                      ;*************************
                      ;
                      ;   W/CRSR-RE/Q/CRSR-LI/A/CRSR-UN
2580:   492C 57 1D 51 UTTAB    .BYTE087,029,081,157,065,017
                      ;        2/CRSR-OB/F1/F2/F3/F4
2600:   4932 32 91 85          .BYTE050,145,133,137,134,138
                      ;F5/F6/F7/F8/L/G
2620:   4938 87 8B 88          .BYTE135,139,136,140,076,071
                      ;I/CTRL.S/CTRL.G/CTRL.X/B
2640:   493E 49 13 07          .BYTE073,019,007,024,066
                      ; C/F/V
2660:   4943 43 46 56          .BYTE067,070,086
                      ;
                      ;LEMEZ TARTALOM
                      ;**************
                      ;
2710:   4946 A9 24    TARTAL   LDA   #$24     ;"$"=FILENEV
2720:   4948 85 02             STA   $02
2730:   494A A9 01             LDA   #$01
2740:   494C A2 02             LDX   #$02
2750:   494E A0 00             LDY   #$00     ;LASD FENN
2760:   4950 20 F9 FD          JSR   FNPAR
2770:   4953 A9 02             LDA   #$02
2780:   4955 A6 BA             LDX   $BA      ;KESZULEKCIM
2790:   4957 A0 00             LDY   #$00
2800:   4959 20 00 FE          JSR   FPAR
2810:   495C A9 00             LDA   #$00     ;LOAD/VERIFY-JELZO
2820:   495E A2 00             LDX   #$00
2830:   4960 A0 40             LDY   #$40     ;KEZDOCIM
2840:   4962 86 5F             STX   $5F
2850:   4964 84 60             STY   $60
2860:   4966 20 D5 FF          JSR   LOAD     ;TARTALOM BETOLTESE PROGRAMKENT
2870:   4969 A5 5F             LDA   $5F      ;SZIMULACIO
2880:   496B A4 60             LDY   $60      ;BASIC-PROGRAM KEZDOCIM
2890:   496D 20 37 A5          JSR   $A537    ;BASICSOROK KOTESE
2900:   4970 AD 00 03          LDA   $0300
2910:   4973 48                PHA
2920:   4974 AD 01 03          LDA   $0301    ;MELEGINDITAS EREDETI MUTATOJA
2930:   4977 48                PHA            ;MENTES
2940:   4978 A9 3D             LDA   #$3D
2950:   497A 8D 00 03          STA   $0300
2960:   497D A9 E3             LDA   #$E3
2970:   497F 8D 01 03          STA   $0301    ;ES RTS
2980:   4982 20 C3 A6          JSR   $A6C3    ;LIST-UTASITAS VEGREHAKJTASA
2990:   4985 68                PLA
```

```
3000:  4986 8D 01 03          STA  $0301
3010:  4989 68                PLA
3020:  498A 8D 00 03          STA  $0300     ;REGI MUTATO VISSZA
3030:  498D 60                RTS            ;VISSZA A BASICBE
                         ;
                         ;SPRITE INVERTALASA
                         ;******************
                         ;
3080:  498E A2 3E    INVERZ   LDX  #62       ;62+1 BYTE
3090:  4990 BD C0 02 IN1      LDA  BLOKK,X ;BYTE BEOLVASASA
3100:  4993 49 FF             EOR  #$FF      ;INVERTALAS
3110:  4995 9D C0 02          STA  BLOKK,X ;VISSZAIRAS
3120:  4998 CA                DEX
3130:  4999 10 F5             BPL  IN1       ;KOVETKEZO BYTE
3140:  499B 60                RTS
                         ;
                         ;SPRITE ELTOLASA
                         ;***************
                         ;
3190:  499C A2 00     JOBB     LDX  #$00      ;ABSZOLUT BYTESZAMLALO
3200:  499E A0 03     RE1      LDY  #$03      ;BYTESZAMLALO A SORBAN
3210:  49A0 18                CLC
3220:  49A1 7E C0 02 RE2      ROR  BLOKK,X ;ELTOLAS
3230:  49A4 E8                INX
3240:  49A5 88                DEY
3250:  49A6 D0 F9             BNE  RE2
3260:  49A8 A9 00             LDA  #$00
3270:  49AA 6A                ROR  A         ;UTOLSO BIT AZ AKKUBA
3280:  49AB 1D BD 02          ORA  BLOKK-3,X
3290:  49AE 9D BD 02          STA  BLOKK-3,X
3300:  49B1 E0 3F             CPX  #63
3310:  49B3 D0 E9             BNE  RE1
3320:  49B5 60                RTS            ;KESZ
                         ;
                         ;
3350:  49B6 A2 3F     BAL      LDX  #63       ;ABSZOLUT BYTESZAMLALO
3360:  49B8 A0 03     LI1      LDY  #$03      ;BYTESZAMLALO A SORBAN
3370:  49BA 18                CLC
3380:  49BB 3E BF 02 LI2      ROL  BLOKK-1,X ;ELTOLAS
3390:  49BE CA                DEX
3400:  49BF 88                DEY
3410:  49C0 D0 F9             BNE  LI2
3420:  49C2 A9 00             LDA  #$00
3430:  49C4 2A                ROL  A         ;UTOLSO BIT AZ AKKUBA
3440:  49C5 1D C2 02          ORA  BLOKK+2,X
3450:  49C8 9D C2 02          STA  BLOKK+2,X
3460:  49CB 8A                TXA
3470:  49CC D0 EA             BNE  LI1
3480:  49CE 60                RTS            ;KESZ
                         ;
                         ;
3510:  49CF A2 3E     FEL      LDX  #62       ;ABSZOLUT BYTESZAMLALO
3520:  49D1 86 02             STX  $02
3530:  49D3 A0 15     OB1      LDY  #21       ;SORSZAMLALO
```

```
3540:  49D5 84 03              STY  $03
3550:  49D7 A6 02              LDX  $02
3560:  49D9 BD 84 02           LDA  BLOKK-60,X ;ELSO BYTE OLVASASA
3570:  49DC 48         OB2     PHA             ;AZ 1-ES KOZBENSO TARBA
3580:  49DD BD C0 02           LDA  BLOKK,X    ;BYTE OLVASASA
3590:  49E0 A8                 TAY             ;A 2-ES KOZBENSO TARBA
3600:  49E1 68                 PLA             ;1-ES KOZBENSO TAR OLVASASA
3610:  49E2 9D C0 02           STA  BLOKK,X    ;BEIRASA EGY BYTE-BA
3620:  49E5 98                 TYA             ;2-ES KOZBENSO TAROLO OLVASASA
3630:  49E6 CA                 DEX
3640:  49E7 CA                 DEX
3650:  49E8 CA                 DEX             ;X REG-1
3660:  49E9 C6 03              DEC  $03        ;KOVETKEZO SOR
3670:  49EB D0 EF              BNE  OB2
3680:  49ED A6 02              LDX  $02
3690:  49EF CA                 DEX
3700:  49F0 86 02              STX  $02        ;KOVETKEZO OSZLOP
3710:  49F2 E0 3B              CPX  #59
3720:  49F4 D0 DD              BNE  OB1
3730:  49F6 60                 RTS
                       ;
                       ;
3760:  49F7 A2 02      LE      LDX  #02        ;ABSZOLUT BYTESZAMLALO
3770:  49F9 86 02              STX  $02
3780:  49FB A0 15      UN1     LDY  #21        ;SORSZAMLALO
3790:  49FD 84 03              STY  $03
3800:  49FF A6 02              LDX  $02
3810:  4A01 BD FC 02           LDA  BLOKK+60,X ;UTOLSO BYTE OLVASASA
3820:  4A04 48         UN2     PHA             ;AZ 1-ES KOZBENSO TARBA
3830:  4A05 BD C0 02           LDA  BLOKK,X    ;BYTE OLVASASA
3840:  4A08 A8                 TAY             ;A 2-ES KOZBENSO TARBA
3850:  4A09 68                 PLA             ;1-ES KOZBENSO TAR OLVASASA
3860:  4A0A 9D C0 02           STA  BLOKK,X    ;ES BEIRASA EGY BYTE-BA
3870:  4A0D 98                 TYA             ;2-ES KOZBENSO TAR OLVASASA
3880:  4A0E E8                 INX
3890:  4A0F E8                 INX
3900:  4A10 E8                 INX             ;BYTE NOVELESE
3910:  4A11 C6 03              DEC  $03        ;KOVETKEZO SOR
3920:  4A13 D0 EF              BNE  UN2
3930:  4A15 C6 02              DEC  $02        ;KOVETKEZO OSZLOP
3940:  4A17 10 E2              BPL  UN1
3950:  4A19 60                 RTS
```

Most pedig a program működésének részletes ismertetése következik.

Ha a bevitelkor nem követtünk el hibát, a program az alábbi képernyőtartalommal jelentkezik:

CÍM

Színek
Következő művelet

24×21-es munkamező

1-es minta
2-es minta

Munkamező:

Ez egy 24×21 négyzetre osztott mező, ami megfelel a sprite-ok felépítésének és nagyon hasonlít a sprite-adatlapra. Ennek minden négyzete a nagyfelbontású sprite egy pontját, ill. a tár 63 byte-jának 1 bitjét jelenti. A sprite-kurzor és a különböző funkciók segítségével a sprite-adatlaphoz hasonlóan bejelölhetjük rajta a bekapcsolt pontokat. Minden sor és oszlop számozott, megadva a pontok Y és X koordinátáit.

A bekapcsolt pontként kiválasztott négyzetek piros színűek lesznek.

1-es minta és 2-es minta

Ez a két ábra a sprite-unk pillanatnyi állapotát mutatja a kiválasztott színben ("F") és a kiválasztott nagyságban ("G"). Az 1-es minta a sprite eredeti, a 2-es pedig a nagyított változatot mutatja, olyan méretben, amekkorában az később a képernyőn meg fog jelenni.

Színek:

Ez egy színskála a rendelkezésünkre álló színekből.

Következő művelet:

Ebben a mezőben további információkat adhatunk meg. A rendelkezésünkre álló utasítások a következők:

B

Kilistázza az utasításkészletet. Tetszőleges billentyű lenyomásával térhetünk vissza a sprite-bevitelre.

Kurzorvezérlő billentyűk

2/Q/W/A

Szintén a sprite-kurzor mozgatását végzik, a billentyűzeten elfoglalt helyüknek megfelelő irányban.

2 = ↑
Q = ←
W = →
A = ↓

F1 ... F8

Ezekhez a billentyűkhöz hozzárendelhetjük az alapszíneket. A minta pontjai ebben a színben jelennek meg.

F

Ezzel az "utasítással" rendeljük hozzá az F1 ... F8 billentyűkhöz a megfelelő színeket. Az ezekkel a billentyűkkel megjelölt mezők a mintában más színűek lesznek.

Figyelem! Az F1/2 (0. háttérszín) a háttérszínek számára van fenntartva.

G

A sprite megnyitását jelenti, amit a 2. mintán kísérhetünk figyelemmel.

I

A sprite invertálása.

V

A jelek a mezőben tetszés szerint eltolhatók. Az információk eközben nem vesznek el, mert az egyik szélen eltűnő jelek a másikon folyamatosan előjönnek.

A V billentyű lenyomása után az eltolás irányát a következő billentyűkkel adjuk meg:

R = jobbra

L = balra

O = fel

U = le

Mindez a mintákon figyelemmel kísérhető.

L

A sprite törlése.

C

A lemez tartalomjegyzékét kilistázza a képernyőre. A listázás a CTRL-billentyűvel lassítható. Bármelyik más billentyű lenyomásával visszatérünk a főprogramba.

CTRL/S és CTRL/G

Ezzel a két utasítással menthetjük ki (CTRL/S) és tölthejük be (CTRL/G) a sprite-unkat.

Adjuk meg a file nevét és – vesszővel elválasztva – a készülékszámot (hibás bevitel esetén az utoljára használt készülék működik). Ezután nyomjuk le a RETURN-t.

Ha a billentyűket tévedésből nyomtuk le, egy sima RETURN-nel visszatérhetünk a bevitelhez. (Ez a többi utasításra, V, F ... stb., is vonatkozik.)

A sprite-unk olyan programfile-ként kerül rögzítésre, amelyet közvetlenül betölthetünk a tárba, vagy szekvenciális file-ként, a már ismertetett módon olvashatjuk be.

CTRL/X

A program befejezése. Ha valóban be akarjuk fejezni a munkát, az ellenőrző kérdésre igennel (J) válaszolunk, töröljük a megmaradt sorokat és nyomjuk le a RETURN-t. Minden más bevitel a főprogram folytatását jelenti.

Reméljük, sikerült megkönnyítenünk a munkát! Sok sikert kívánunk a program alkalmazásához.

4.3.2 A sprite-ok jellemzőinek programozása

A 3.5 fejezetben részletesen foglalkoztunk a sprite-ok definiálásának, bekapcsolásának és változtatásának elméletével. Itt az idő, hogy megtanuljuk ezeket a dolgokat BASIC-ben megvalósítani.

Egy kis példaprogramon keresztül fogjuk bemutatni a variációs lehetőségeket, és bemutatjuk a sprite-ok tipikus alkalmazási területét, az animációt, vagyis a mozgó kép készítését.

A példaprogramot tetszés szerint módosíthatjuk, de a POKE-utasításokkal óvatosan bánjunk. Mindig gondoljuk meg jól, mit idézünk elő egy esetleges változtatással. Ha valamit mégis elrontottunk, a gép kikapcsolásával segíthetünk magunkon.

Térjünk a tárgyra:

```
1 REM *** P 25 ***
2 REM
1000 REM ******************
1010 REM **              **
1020 REM ** SPRITE PELDA **
1030 REM **              **
1040 REM ******************
1050 REM
1060 V=53248:REM A VIC KEZDOCIME
1070 POKE V+32,0:POKE V+33,0:REM KERET/H.TER FEKETE
1080 REM
1090 REM SPRITE-MINTA A 13-AS BLOKKBA:
1100 REM *****************************
1110 A1=13*64:REM 13-AS BLOKK CIME
1120 FOR X=0 TO 62
1130 READ DT:POKE A1+X,DT
1135 REM ADATOK BEOLVASASA ES BEIRASA
1140 NEXT X
1150 REM ELSO SPRITE:
1160 DATA 000,007,000
1170 DATA 056,013,128
1180 DATA 025,031,224
1190 DATA 126,031,204
1200 DATA 025,012,252
1210 DATA 056,007,000
1220 DATA 000,014,000
1230 DATA 000,014,000
1240 DATA 000,015,128
1250 DATA 000,014,192
1260 DATA 000,030,096
1270 DATA 000,062,048
1280 DATA 000,046,000
1290 DATA 000,079,000
1300 DATA 000,155,128
1310 DATA 001,025,192
1320 DATA 002,024,096
1340 DATA 127,024,120
1350 DATA 188,024,000
1360 DATA 036,030,000
1370 REM
1380 REM SPRITE-MINTA A 14-ES BLOKKBA:
```

```
1390 REM *****************************
1400 A2=14*64:REM A 14-ES BLOKK CIME
1410 FOR X=0 TO 62
1420 READ DT:POKE A2+X,DT
1430 REM ADATOK BEOLVASASA ES BEIRASA
1440 REM MASODIK SPRITE
1450 DATA 000,003,010
1460 DATA 000.013,132
1470 DATA 058,031,224
1480 DATA 124,031,170
1490 DATA 058,015,250
1500 DATA 000,007,000
1510 DATA 000,014,000
1520 DATA 000,014,000
1530 DATA 000,014,000
1540 DATA 000,015,000
1550 DATA 000,015,128
1560 DATA 000,014,160
1570 DATA 000,030,000
1580 DATA 000,047,000
1590 DATA 000,077,128
1600 DATA 000,141,128
1610 DATA 001,057,128
1620 DATA 003,097,128
1630 DATA 127,113,128
1640 DATA 220,025,128
1650 DATA 036,001,224
1660 POKE 2046,A1/64:REM 6.SPRITE A 13-AS BLOKKBOL
1670 POKE V+21,2↑6:REM 6. SPRITE BE
1680 POKE V+39+6,1:REM 6.SPRITE -FEHER
1690 POKE V+27  ,0:REM SPRITE AJHATTER ELOTT
1700 POKE V+28  ,0:REM NORMAL SPRITE-SZIN
1710 POKE V+23,2↑6:REM SPRITE NAGYITASA Y-IRANYBAN
1720 POKE V+29,2↑6:REM SPRITE NAGYITASA X-IRANYBAN
1730 POKE V+2*6+1,101:REM 6. SPRITE Y-KOORD.
1740 POKE V+16,0:REM X-KOORD. FELSO BIT TORLESE
1750 REM
1760 PRINT CHR$(30) CHR$(147):REM ZOLD+KEPERNYOT.
1770 FOR X=1 TO 10
1780 PRINT:REM 10 URES SOR
1790 NEXT X
1800 PRINT CHR$(18);:REM RVS ON
1810 FOR X=1TO 40
1820 PRINT " ";:REM 40 INVERZ ZOLD SZOKOZ
1830 NEXT X
1840 REM
1850 REM
1860 REM **********
1870 REM **      **
1880 REM ** FUTAS **
1890 REM **      **
1900 REM **********
1910 REM
1920 G=10   :REM SEBESSEG
```

```
1930 F=1    : REM KEZDOSZIN
1940 FOR X=1 TO 400 STEP G
1950 F=F+.3:REM SZINVALTAS
1960 IF F=16 THEN F=1
1970 REM ----------------------------
1980 POKE V+39+6,F:REM SZINVALTAS
1990 POKE 2046,A1/64:REM 6.SPRITE A 13-AS BLOKKBOL
2000 KO=X
2005 IF X>255 THEN KO=X-256:POKE V+16,2↑6:GOTO 2020
2010 POKE V+16,0:REM X-KOORD. FELSO BIT TORLESE
2020 POKE V+2*6,KO:REM SPRITE MOZGATASA
2030 FOR Y=1 TO 40:NEXT Y:REM VARAKOZAS
2040 REM ----------------------------
2050 POKE 2046,A2/64:REM 6. SPRITE A 14-ES BLOKKBOL
2060 KO=X+G/2
2065 IF KO>255 THEN KO=KO-256:POKEV+16,2↑6:GOTO2080
2070 POKE V+16,0:REM X-KOORD.   FELSO BYTE TORLESE
2080 POKE V+2*6,KO:REM SPRITE MOZGÁTASA
2090 FOR Y=1 TO 40:NEXT Y:REM VARAKOZAS
2100 REM ----------------------------
2110 NEXT X
```

Az 1060-as sorban először is megadjuk a V változó értékét, ami most is 53248, azaz a VIC kezdőcíme lesz. Ezt minden sprite-okkal dolgozó vagy grafikus program előtt meg kell adnunk. Ezután a háttér és a keret színét feketére változtatjuk (1070-es sor).

a) *A sprite-minta*

A program 1100-as sorától kezdődő rutint, ami a DATA-sorokból beolvassa a tár megfelelő helyére a sprite-ot jellemző 63 byte-ot, az előző fejezetből már ismerjük. Az egyik sprite mintája a 13., a másiké (1400-as sortól) a 14. blokkba kerül. A blokkok kezdőcímét a már ismert módon kiszámítjuk: kezdőcím = blokkszám*64, majd az A1 és A2 változóba töltjük (1100-as és 1400-as sor).

Előállítottunk és tároltunk tehát két sprite-ot, amelyek, mint látni fogjuk, nagyon hasonlóak. Mindkét esetben egy pipázó öregúrról van szó, aki a kutyájával sétál, előttük pedig egy madár repül. A két sprite egy ember, a pipafüst és a madár mozgásának két fázisát rögzíti.

Külön is megnézhetjük őket, ha a 2030-as és 2090-es sorba egy-egy STOP-utasítást írunk.

A program célja nem az, hogy a két sprite-ot egyszerre jelenítse meg a képernyőn, hanem annak bemutatása, hogy a sprite-ok alakjának csekély eltérése és váltakozó megjelenítése miképpen hozza létre a mozgást.

b) *Blokkmutató*

A sprite száma: 6. Először annak a blokknak a számát kell megadnunk, ahonnan a 6-os sprite mintáját venni akarjuk. Ez először a 13-as blokk lesz, tehát a 3.5.3 fejezetben leírtak szerint a video-RAM utolsó 8 byte-ja közül a hatodikban 13-at kell írnunk (1660-as sor). Ez sprite-minta mutatója, más néven vektora.

c) *Bekapcsolás*

Ahhoz, hogy a sprite a képernyőn is megjelenjen, be is kell kapcsolnunk. Ez az 1670-es sorban történik, amellyel a megfelelő VIC-regiszterbe (21-es) beírjuk a 2↑6 = 64 értéket. Ezzel bekapcsoltuk a 6. bitet, ami egyben a sprite bekapcsolását jelenti.

d) *Színek*

Könnyen előfordulhat, hogy hiába kapcsoltuk be a sprite-unkat, az nem jelenik meg a képernyőn, mert egyelőre a képernyőablakon kívül tartózkodik. Ezzel most még ne foglalkozzunk, inkább adjuk meg a sprite színét. Ehhez a 39–46-os VIC-regisztereket kell igénybe vennünk.

Az 1680-as sorban a nyolc regiszter közül a hatodikba (39+6 = 45) beírjuk a fehér szín kódját (1) és ezzel meghatározzuk a 6-os sprite számára a fehér színt.

e) *Elsőbbségi viszony*

Az 1690-es sorban döntjük el, hogy a sprite a háttérjelek előtt vagy azok mögött mozogjon-e. Más szóval a később megjelenő zöld vonal takarja-e el a sprite-ot, vagy fordítva. Az utóbbi esetet választva kikapcsoljuk a 27-es VIC-regiszter megfelelő bitjét. Ha a másik esetet is ki akarjuk próbálni, módosítsuk a sort a következők szerint:

```
1690 POKE V+27, 2 ↑ 6
```

Nézzük át még egyszer a 3.5.4.3 fejezetet!

f) *Többszínűség*

Mindkét sprite-unkat egyszínűként terveztük és hoztuk létre. Emiatt csak ebben az üzemmódban kapcsolhatók be. Ez a tény tehát, mint látjuk, már nagyon korán, szinte a tervezés pillanatában eldől.

g) *Nagyítás*

El kell döntenünk, hogy a figuránkat eredeti méretében, vagy nagyítva kívánjuk-e viszontlátni. A szemléletesség kedvéért mi a mindkét irányú (X–Y) nagyítást választottuk, amelyhez az 1710-es és 1720-as sorokban a 23-as és 29-es VIC-regiszterek megfelelő bitjeit bekapcsoltuk (ld. a 3.5.4.2 fejezetet!).

Próbáljuk ki a nagyítás különböző variációit, és nézzük meg az egészet eredeti nagyságban is.

Kísérletezzünk, gyakoroljunk!

h) *A sprite helye a képernyőn*

Eljutottunk végre oda, hogy a sprite-unkat a képernyőre vigyük. Adjuk meg a képernyő azon pontjának koordinátáit, ahol meg akarjuk jeleníteni. Később ezek az adatok majd megváltoznak, de a teljesség kedvéért már itt is megemlítjük.

A 3.5.4.1 fejezetből már tudjuk, hogy a sprite-ok helyének kijelölése a 0–16-os VIC-regiszterekkel történik. A 6-os sprite esetében a 12-es (X koordináta alsó byte-ja) és a 13-as (Y koordináta) regiszterekre van szükségünk. Az X kordináta felső bitjét most is a 16-os regiszter megfelelő (6) bitje adja (1730–1740-es sor). Ebben az állapotban a sprite-unk már látható. Elvégeztünk tehát minden olyan tennivalót, ami egyetlen sprite-tal dolgozó programból sem hiányozhat. Láthatjuk, hogy bármilyen kényelmes eszköz áll is rendelkezésünkre, a sprite-ok kezelése egyáltalán nem könnyű dolog. Idővel azonban olyan jártasságra teszünk szert, hogy magunk is megtaláljuk az egyszerűsítés módjait. Különböző dolgokat lerövidíthetünk, egyeseket akár el is hagyhatunk.

Folytassuk a program vizsgálatát!

Az 1750-es sortól kezdődő programrész következik, ami az ún. látható dolgokért felelős.

Első lépésként töröljük a képernyőt (1760-as sor), majd elindítjuk a zöld vonalat. Ezután egy FOR...NEXT-ciklust indítunk 1-től 4000-ig, ami az X koordináta értékét növeli a G változóban tárolt léptékkel.

A ciklus lefutása alatt sok minden történik.

Először is minden átfutáskor nő az F értéke, ami a színkódot tárolja. Ennek megfelelően minden egész érték elérésekor színváltás történik. F=16 után a számolás újra 1-től indul (1960-as sor). Az 1980-as sorban a színregiszterbe (39+6 = 45) beírjuk F egész értékét. (Az egész érték minden harmadik lefutás után változik.)

A következő programrész feladata a sprite-minta változása, ami a mozgást hozza létre.

A két sprite-mintát a 13-as és 14-es blokkban tároltuk. Ezeket váltakozva kell előhívni, ezért az 1990-es sorban a 13-as, a 2050-es sorban a 14-es értéket írtuk be mutatóként a 2046-os tárrekeszbe.

(A 2046-os byte a video-RAM utolsó nyolc byte-ja közül a hatodik, amelybe annak a blokknak a számát kell beírnunk, amelyből a 6-os sprite mintáját venni akarjuk.)

Ahhoz, hogy a mozgást folyamatosnak lássuk, az X koordinátákat minden átfutáskor egy vagy több ponttal meg kell növelni. Az Y koordináta állandó marad. Az X koordináta értékét X változóban tároljuk. A KO változót (2000-es sor) közbenső tárolóként használjuk arra az esetre, ha az X értéke meghaladná a 255-öt. Ekkor ugyanis igénybe kell vennünk az ún. felső bitet a 16-os regiszterből és az

ILLEGAL QUANTITY ERROR = (illegális mennyiség)

hibaüzenet elkerülése érdekében KO értékét 256-tal csökkenteni kell. Ezután meghatározzuk az alsó byte-ot (2020-as sor), és ezzel meghatároztuk a sprite helyét.

A program harmadik részében egy várakozási ciklus után a folyamat megismétlődik (2060-as sor).

Az ismertetett példában a sprite-ok kezelésének néhány technikai megoldását mutattuk be. Van azonban néhány dolog, amiről eddig még említést sem tettünk. Ilyen többek között a sprite-ok ütköztetése, több sprite egyidejű megjelenítése, vagy a többszínű sprite-ok. Először ezekről beszélünk.

Tudjuk, hogy a többszínű üzemmódban egy sprite-ot négy színből állíthatunk

össze, emiatt viszont felére csökken az X irányú felbontás és minden pont kétszeres szélességű lesz. A tárban mindig 2 bit határozza meg a képernyő egy dupla szélességű pontjára vonatkozóan a pont színének származási regiszterét. Ezek a VIC 37/38-as (0-ás és 1-es többszín-regiszterek), valamint a normál sprite–szín-regiszterek (39–46) lehetnek. Ha a pontot meghatározó mindkét bitet kikapcsoljuk, ezen a helyen a háttérjelek érvényesülnek (3.5.3 fejezet).

Következő témánk a sprite-ok ütköztetése. Két típusát különböztetjük meg: sprite– sprite-ütközés és sprite–háttérjel-ütközés. Az előzőt a 30-as, az utóbbit a 31-es VIC-regiszter rögzíti oly módon, hogy az ütközésben részt vevő sprite-hoz tartozó bit bekapcsolódik. Ezzel kapcsolatban olvassuk át még egyszer a 3.5.5.4 fejezetet.

Következő példaprogramunkkal a sprite– sprite közti ütközést szemléltetjük.

```
1 REM *** P 26 ***
2 REM
100 REM ***********************
110 REM **                   **
120 REM ** SPRITE UTKOZTETES **
130 REM **    (TOBBSZINU)    **
140 REM ***********************
150 REM
160 V=53248:REM A VIC KEZDOCIME
170 A1=11*64:REM 11-ES BLOKK
180 FOR X=0 TO 62
190 READ DT:POKE A1+X,DT
195 REM SPRITE MINTA BEOLVASASA ES BEIRASA
200 NEXT X
210 POKE 2040,11:POKE 2042,11
215 REM 0. ES 2. SPRITE MUTATOJA A 11-ES BLOKKRA
220 POKE V+28,2↑0 OR 2↑2:REM 0. ES 2. SPRITE TSZ.
230 POKE V+27,2↑0:REM 2. SPRITE A HATTER ELOTT
240 POKE V+29,2↑0 OR 2↑2:POKE V+23,2↑0 OR 2↑2
245 REM MINDKET SPRITE NAGYITVA
250 POKE V+21,2↑0 OR 2↑2:REM MINDKET SPRITE BE
260 POKE V+37,7:REM 0. SPRITE TSZ.=SARGA
270 POKE V+38,6:REM 1. SPRITE TSZ.=KEK
280 POKE V+39,5:POKE V+41,8:REM EGYEDI SZINEK
290 POKE V+16,0:REM X-KOORD.-FELSO BITEK TORL.
300 POKE V+1,100:POKE V+5,100:REM Y-KOORD.-K BEALL.
310 X2=255:REM 2. SPRITE KIINDULASI KOORD.(X)
320 POKE V+0,0:POKE V+4,X2:REM X-KOORD.-K BEALL.
330 POKE V+30,0:REM UTKOZESI REG. TORLESE
340 FOR X0=1 TO 255:REM 0. SPRITE X-KOORDINATAI
350 X2=X2-1:REM 2. SPRITE X-KOORD. CSOKKENTESE
360 POKE V+0,X0:POKE V+4,X2
365 REM 0/2 SPRITE X-KOORD. (ALSO BYTE)
370 POKE V+1,40*SIN(X0/30)+100
375 REM MOZGAS SZINUSZ-PALYAN
```

```
380 POKE V+5,40*COS(X0/30)+100
385 REM MOZGAS KOSZINUSZ-PALYAN
390 IF PEEK(V+30)=(2↑0 OR 2↑2) THEN GOTO 420
400 NEXT X0
410 REM
420 REM UTKOZESI ALPROGRAM
425 REM ******************
430 REM
440 FOR X=1 TO 255
450 POKE V+39,X:POKE V+41,X:REM SPRITE-SZIN VALTAS
460 NEXT X
470 POKE V+0 ,0:REM
480 POKE V+30,0:REM UTKOZESI REGISZTER TORLESE
490 REM
500 REM SPRITE MINTA
510 REM
520 DATA 008,000,128
530 DATA 010,002,128
540 DATA 002,138,000
550 DATA 064,136,005
560 DATA 080,168,021
570 DATA 084,032,093
580 DATA 117,033,125
590 DATA 125,101,253
600 DATA 127,103,253
610 DATA 123,103,173
620 DATA 123,103,173
630 DATA 123,103,173
640 DATA 123,103,173
650 DATA 123,103,173
660 DATA 123,103,173
670 DATA 123,103,173
680 DATA 123,103,173
690 DATA 127,103,253
700 DATA 095,101,253
710 DATA 021,097,117
720 DATA 005,032,084
```

Először is el kell végeznünk a sprite létrehozásához szükséges műveleteket. Ebben a programban csak egy sprite-mintával dolgozunk, amit a 11-es blokkban rögzítünk (170–200-as sor).

A képernyőn egyszerre két sprite-ot fogunk megjeleníteni, amelyek a 0-ás és 2-es sorszámot kapják. Alakjuk egyforma. Ezt a 210-es sor határozza meg, amelyben a 0-ás és 2-es sprite mutatójaként egyaránt 11-et adunk meg. A többi paraméterrel ettől függetlenül dolgozhatunk. Ezt láthatjuk például a 230-as sorban, ahol csak a 0-ás sprite-ra vonatkozóan írtuk elő a háttérjelekkel szembeni elsőbbséget, valamint a 280-as sorban, ahol a két sprite számára különböző színt határoztunk meg.

Mint mondtuk, többszínű sprite-okkal akarunk dolgozni, ezért a DATA-sorokban így adtuk meg az értékeket. Mindezt a VIC-kel is közölnünk kell. Rögtön felmerül egy probléma, amivel eddig még nem találkoztunk. Eddig minden regiszterben csak egy bitet kellett bekapcsolnunk. Hogy oldható meg ez egyszerre két bittel? Nos, erre van az OR logikai művelet, amelyet részletesen a 2. fejezetben ismertettünk. A 2↑0 (0. bit bekapcsolása) és a 2↑2 (2. bit bekapcsolása) műveleteket tehát OR-utasítással kell összekapcsolnunk.

Új dologgal találkozunk a 260–280-as sorokban is. Itt írjuk be a megfelelő regiszterekbe a színkódokat. Ne felejtsük el, hogy csak a 3-as szín (3-as színcsatorna) lehet sprite-onként különböző.

A 290–320-as sorokkal visszük be a sprite-jainkat a képernyőablak területére. Erre valójában nincs is szükség, mivel minden paraméter (a 16-os regiszter bitjeinek kivételével) a következő mozgásciklusban is beállításra kerül. Van azonban egy körülmény, ami mégis szükségessé teszi. A sprite-okat ütközés szempontjából akarjuk vizsgálni és a bekövetkező ütközést valamilyen módon jelezni is akarjuk. Könnyen előfordulhat, hogy a sprite-ok már keletkezésük pillanatában érintkeznek, hiszen ilyenkor minden koordinátaérték 0. Ezt a számítógép ütközésnek minősíti és a program közvetlenül az indítás után elágazik a megszakítási rutinba, holott a sprite-ok még meg sem jelentek a képernyőn. Ahhoz, hogy ezt elkerüljük, a sprite-okat először el kell egymástól különítenünk.

Ezután – és ez nagyon fontos – vissza kell állítanunk a 30-as regiszter eredeti állapotát, vagyis a bekapcsolt biteket ki kell kapcsolni. Ez azért nagyon fontos, mert mint annak idején már elmondtuk, egy ütközésre vonatkozó bejegyzés (a megfelelő bit bekapcsolása) mindaddig a regiszterben marad, míg azt ki nem töröljük, még akkor is, ha több ütközés nem történik.

Ezek után továbbmehetünk. Most is, mint az előző példában a sprite-ok mozgását egy FOR...NEXT-ciklussal hoztuk létre. Az X0 ciklusszámláló folyamatosan nő 1-től 255-ig és a 0. sprite X koordinátáját adja (340-es sor). Az X2 változó a 2. sprite X koordinátáját tartalmazza és értéke a ciklus során 250-től 1-ig csökken (350-es sor). A kiszámított X koordinátákat a 360-as sor írja be a megfelelő (V+0 és V+4) regiszterekbe. Ahhoz, hogy a 0-ás sprite szinusz-, a 2-es sprite pedig koszinuszgörbe mentén mozogjon, az Y koordinátákat a megfelelő képletből kell kiszámítani (370-es, 380-as sor). Gyakorlásképpen nyugodtan változtassuk meg a paramétereket! A 40 az amplitúdót, a 30 a hullámhosszt, a 100 pedig a görbe Y irányú eltolását jelenti.

Most egy újabb érdekes téma következik: az ütközés lekérdezése.

A 390-es sorban kiolvassuk a 30-as (sprite–sprite-ütközési) regiszter tartalmát. Amennyiben ebből az derül ki, hogy a 0-ás és 2-es sprite ütközött (a 0/2 bit értéke 1), a program elágazik a 420-as sorban kezdődő ütközési alprogramba. A képernyőn egy kis színeffektus jön létre, majd a sprite-ok eltávolodnak egymástól és az ütközési regiszter törlődik. (A regiszter törlése előtt mindenképpen szüntessük meg a sprite-ok érintkezését a már említett okból.)

Ezzel minden együtt van, ami a sprite-ok ütközésvizsgálatához szükséges.

Úgy gondoljuk, elég szemléletes leírást adtunk a sprite-ok programozásáról. Most már csak a gyakorlás van hátra. Lehetőségeinknek csak a fantáziánk szab határt.

Jó munkát és a kísérletezéshez sok sikert kívánunk!

4.4 A jelkészlet programozása

A jelkészlet módosítása, bár roppant lehetőségeket rejt magában, meglehetősen mostoha téma. Nagyon kevés olyan igényes játék van, amelyik nem használja ki a számítógép ezen tulajdonságait, sőt soknak a lényege éppen erre épül. Szinte grafikai képességként kezelhető és majdnem egy szinten említhető a nagyfelbontású grafikával, megelőzve azt a feldolgozási sebesség tekintetében. Kb. 8–9-szerese a nagyfelbontású grafika feldolgozási sebességének. Az egyedi jelkészlet létrehozásának lehetősége a C 64-est rendkívül jól illeszthetővé tette. Megszűnt az a hátrány, ami sok számítógépnél abban nyilvánul meg, hogy betűkészlete eltér az adott országban használatos betűkészlettől. Előállíthatunk az amerikain kívül német, svéd, vagy akár ciril, japán vagy görög betűkészletet is. Az egyes jelkészletek cserélhetők.

Nem lebecsülendő tehát számítógépünk ezen képessége, éppen ezért ebben a fejezetben a jelkészlet módosításának programtechnikai megvalósításával ismerkedünk meg. Ez jóval bonyolultabb feladat, mint a sprite-ok programozása, és nem térhetünk ki a gépi kódú programozás elől. (Kivéve, ha esetleg rendelkezünk valamilyen BASIC-bővítéssel.) Emiatt a kezdők ezt a részt egyelőre hagyják ki. Mielőtt komolyabban elmélyednénk a témában, ajánlatos még egyszer átolvasni a 3.6 fejezetet.

Azt már tudjuk, hogy minden jel definíciója 8 byte-ból áll. Ez jelenként egy

8×8-as pontmátrixot ad. Egyidejűleg 512 különböző jel tárolható, de ebből csak legfeljebb 256 jelenhet meg egyszerre a képernyőn. Az 512 különböző jel 4 k-t foglal el a tárból és normál esetben a $D000–$DFFF (53218–57343) tartományban, az ún. jelkészlet-ROM-ban található.

Ezt a VIC 24-es regiszter és a CIA2 0. regiszterének 0/1 bitjeivel módosíthatjuk (címek: $D018=53272 és $DD00=56576).

Röviden ennyit ismétlésként.

Jelkészlet készítésénél két esetet kell megkülönböztetnünk. Az első esetben a meglevő jelkészletnek csak néhány elemét változtatjuk meg (különleges jelek), a másik esetben viszont teljesen új jelkészletet hozunk létre.

4.4.1 Néhány jel módosítása

Ha a jelkészletben bármiféle változtatást akarunk végrehajtani, mindenekelőtt át kell másolnunk a ROM-ból egy erre a célra kijelölt RAM-tartományba, majd módosítanunk kell a jelkészlettár kezdőcímét a 3.3.2 fejezetben leírt módon. Itt jegyezzük meg, hogy a 3.3 fejezetet teljes egészében és biztosan érteni kell. Legjobb újra elolvasni.

A RAM olvasható és írható is, ezért a szükséges változtatásokat itt tudjuk elvégezni.

Mindez most egyszerűnek és rövidnek tűnhet, de még jó néhány további ismeretet és komoly programozástechnikai jártasságot feltételez. Tudnunk kell például, hogy a jelkészlettár BASIC-ből nem olvasható, így nem is másolható. Ennek az az oka, hogy a $D000–$DFFF (53248–57343) tártartomány az egész be-/kiviteli (I/O) tartományt átfogja és így először a CPU 1. regiszterének módosításával be kell kapcsolni. A be-/kiviteli tartomány, amit az operációs rendszer megszakítási rutinja is használ, BASIC-ben használhatatlanná válik. Gépi nyelven a megszakítás a CPU megszakításjelzőjének (interruptflag) beállításával megakadályozható.

A következőkben közreadunk egy gépi nyelvű programot, ami elvégzi a másolást és a jelkészlettár eltolását. Ez utóbbi BASIC-ben is megoldható, de így egyszerűbb.

```
                          ;   *** P 27 ***
                          ;
                          ;JELKESZLET ELTOLASA
                          ;*******************
                          ;
C800                            *=   $C800     ;KEZDOCIM
C800               V            =    53248     ;VIDEOVEZERLO BAZISCIME
C800               JEL          =    53248     ;JELKESZLET BAZISCIME
C800               CEL          =    $3000     ;MASOLAS KEZDOCIME
C800 78            START   SEI                 ;MEGSZAKITAS TILTASA
C801 A5 01                 LDA     $01         ;CPU-REG 1
C803 48                    PHA                 ;MENTES
C804 29 FB                 AND     #%11111011 ;2-ES BIT TORLESE
C806 85 01                 STA     $01         ;JELKESZLETTAROLO KIIKTATASA
C808 A9 D0                 LDA     #>JEL
C80A 85 03                 STA     $03         ;FORRASCIM FELSO BYTE
C80C A9 30                 LDA     #>CEL
C80E 85 05                 STA     $05         ;CELCIM FELSO BYTE
C810 A0 00                 LDY     #$00
C812 84 02                 STY     $02
C814 84 04                 STY     $04         ;ALSO-BYTES =00
C816 A2 20                 LDX     #$20        ;A 4K BYTE SZAMLALOJA
C818 B1 02         COPIE   LDA     ($02),Y ;BYTE TOLTESE
C81A 91 04                 STA     ($04),Y ;ES MASOLASA
C81C C8                    INY
C81D D0 F9                 BNE     COPIE       ;256-SZOR
C81F E6 03                 INC     $03
C821 E6 05                 INC     $05         ;FELSO-BYTE NOVELESE
C823 CA                    DEX
C824 D0 F2                 BNE     COPIE       ;4K MASOLASA
C826 68                    PLA
C827 85 01                 STA     $01         ;JELKESZLET TAROLO BE-KI
C829 AD 18 D0              LDA     V+24        ;JELKESZLET TAR CIME
C82C 29 F1                 AND     #%11110001
C82E 09 0C                 ORA     #%00001100
C830 8D 18 D0              STA     V+24        ;$3000 CIMRE
C833 58                    CLI                 ;MEGSZAKITAS ISMET ENG
C834 60                    RTS                 ;VISSZA A BASICBA
```

A program BASIC-betöltő listája:

```
1 REM *** P 27 B ***
2 REM
100 FOR I= 51200 TO 51252
110 READ X:POKE I,X:S=S+X:NEXT I
120 DATA 120,165,001,072,041,251
125 DATA 133,001,169,208,133,003
130 DATA 169,048,133,005,160,000
135 DATA 132,002,132,004,162,032
140 DATA 177,002,145,004,200,208
145 DATA 249,230,003,230,005,202
150 DATA 208,242,104,133,001,173
155 DATA 024,208,041,241,009,012
160 DATA 141,024,208,088,096
170 IF S<>5884 THEN PRINT"HIBA A DATA-SOROKBAN !!":END
180 PRINT"RENDBEN !"
```

Töltsük be programot, és indítsuk el RUN-nal. Ezután SYS 51200-zal az eredeti jelkészlettárat máris áthelyeztük. Ha van monitorprogramunk, a gépi nyelvű programot közvetlenül is begépelhetjük, tárolhatjuk, majd a LOAD "név", 8,1 paraccsal újból betölthetjük.

A 3.6 fejezetből már ismerjük a jelek 8×8-as (ill. 4×8-as) felépítését és 8 byte-ban való tárolását.

Egy jel mintájának kicserélése több lépésből áll. Először is meg kell tervezni az új mintát. Ez, a sprite-okhoz hasonlóan, egy ún. jeltervező lapon történik (ld. Függelék). A tervezőlap minden pontsora a tárban 1 byte-ot, és minden bejelölt pontja egy bekapcsolt bitet jelent. Ezt az alábbi ábrán szemléltetjük:

Bit:	7	6	5	4	3	2	1	0
0. byte :	.	.	.	*	*	*	.	.
1. byte :	.	*	*	*	.	.	.	.
2. byte :	*	*	.	.	*	*	*	*
3. byte :	*	*	.	.	.	.	.	.
4. byte :	*	*	.	.	*	*	*	*
5. byte :	.	*	*	*	.	.	.	.
6. byte :	.	.	.	*	*	*	.	.
7. byte :	.	.	.	.	.	.	.	.

A byte-ok tartalma a következő:

0. byte: %0001 1100=$1C=028

1. byte: %0111 0000=$70=112

2. byte: %1100 1111=$CF=207

3. byte: %1100 0000=$C0=192

4. byte: %1100 1111=$CF=207

5. byte: %0111 0000=$70=112

6. byte: %0001 1100=$1C=028

7. byte: %0000 0000=$00=000

Rögtön adódik a kérdés: melyik jelet helyettesítsük és melyik jelkészletben? (Ld. 3.6 fejezet.)

Tételezzük fel, hogy a nagybetű/grafikus jelek üzemmódban elérhető font (£) jelet akarjuk az általunk megszerkesztett commodore (C=) jellel helyettesíteni. Először is meg kell állapítanunk, hogy a £-jelet rögzítő 8 byte a tárban hol helyezkedik el. Használjuk fel a 3. fejezetből már ismerős képletet:

cím = kezdőcím+8*képernyőkód.

A kezdőcím alapvetően attól függ, hogy hová toltuk el a jelkészlettárat. Mivel például a $3000–$3FFF (12288–16383) tartományában vagyunk és csak a nagybetű/grafikus jelkészletet akarjuk módosítani, az első érték már adott. Marad a képernyőkód, amit a Függelékben levő táblázatból kereshetünk ki. A normál £-jel képernyőkódja 28 ($1C), az inverz változaté 28+128 = 256. Helyettesítsük be az értékeket:

cím = 12288+8*28 = 12512 = $30E0.

Ezzel megállapítottuk, hogy a normál £-jel mintája a tár $30E0–$30E7 (12512–12519) tartományában van. Ha most ide beírjuk az általunk tervezett különleges jel mintáját, a £-billentyű lenyomásával a képernyőre egy C=-jel kerül.

A következő kis BASIC-program ezt valósítja meg, természetesen csak a jelkészlettár átmásolása és eltolása után:

```
1 REM *** P 28 ***
2 REM
10 REM JELMODOSITAS
20 FOR X=0 TO 7
30 READ DT:REM A BITMINTA BEOLVASASA
40 POKE 12288+8*28+X,DT:REM ES BEIRASA
50 NEXT X
60 DATA 028,112,207,192,207,112,028,000
```

A P29-es programmal, a P27 B és a P28 programok összefűzésével, a futtatható változatot is bemutatjuk.

```
1 REM *** P 29 ***
2 REM
10 GOSUB 100:SYS 51200:REM JELKESZLETTAR. ELTOLASA
15 REM JELMODOSITAS
20 FOR X=0 TO 7
30 READ DT:REM A BITMINTA BEOLVASASA
```

```
40 POKE 12288+8*28+X,DT:REM ES BEIRASA
50 NEXT X:END
60 REM ------------------------
100 FOR I= 51200 TO 51252
110 READ X:POKE I,X:S=S+X:NEXT I
120 DATA 120,165,001,072,041,251
125 DATA 133,001,169,208,133,003
130 DATA 169,048,133,005,160,000
135 DATA 132,002,132,004,162,032
140 DATA 177,002,145,004,200,208
145 DATA 249,230,003,230,005,202
150 DATA 208,242,104,133,001,173
155 DATA 024,208,041,241,009,012
160 DATA 141,024,208,088,096
170 IF S<>5884 THEN PRINT"HIBA A DATA-SOROKBAN !!":END
180 PRINT"RENDBEN !":RETURN
190 REM ------------------------
200 DATA 028,112,207,192,207,112,028,000
```

Mielőtt a programot elindítjuk, vigyünk néhány £-jelet a képernyőre, hogy lássuk az eredményt. Ha ezután átkapcsolunk a kis-/nagybetűs jelkészletre, meggyőződhetünk arról, hogy ebben az esetben a £-jel definíciója a tár más tartományából származik.

Ugyanez természetesen minden más jellel is elvégezhető, így létrehozhatunk egy saját, különleges és egyedi jelkészletet, amit a legkülönbözőbb célokra használhatunk.

4.4.2 A teljes jelkészlet módosítása

Ha az egész jelkészletet ki akarjuk cserélni, nem kell a másolás bonyolult műveletével foglalkoznunk. Ebben az esetben teljesen elegendők a BASIC nyújtotta lehetőségek. Az új jelkészletet egyszerűen betöltjük a kiválasztott tártartományba, majd közölni kell a VIC-kel, hogy az egyes jelek mintáját ezentúl innen vegye.

Mielőtt hozzáfognánk a jelkészlet létrehozásához, el kell döntenünk, hogy milyen betűkészletet (cirill, görög ... stb.), írott vagy nyomtatott betűket, esetleg valamilyen egyéb jeleket akarunk-e készíteni.

Egy jelkészlet sok különböző jelből áll, amiket egyenként kell tervezni és monitorprogrammal, vagy BASIC-ből POKE-utasításokkal be kell vinni a tárba. Ez nem kis munka, sok programozó kételkedik is a megvalósíthatóságában. Mi segítségképpen közreadunk egy programot, ami nagyon hasonlít a már ismertetett sprite-tervező programunkra (P27). A szükséges magyarázatokat REM-es sorokban adjuk meg.

```
1 REM *** P 30 ***
2,REM
10 REM *******************************
20 REM **                           **
30 REM ** JELTERVEZO SEGEDPROGRAM **
40 REM **                           **
50 REM *******************************
60 REM
70 REM ELOKESZITES
80 REM ***********
90 GOSUB 10000:REM GEPI ALPROGRAMOK BEOLVASASA
100 POKE 53280,0:POKE 53281,0
105 REM HATTER ES KERET FEKETE
110 POKE 763,0:POKE 650,255
115 REM UZEMMOD=0;MINDEN JEL ISMETLO
120 POKE 45,0:POKE 46,80:RUN 130
125 REM BASIC TAROLO VEGE = $5000
130 REM
140 REM GEPI KODU ALPROGRAMOK
150 REM
160 IN%=18432 :REM ELOKESZITES
170 PU%=18633 :REM PONT BEJELOLESE
180 NE%=18586 :REM KOORDINATARENDSZER
190 LA%=18487 :REM JELKESZLET TOLTESE
200 SP%=18515 :REM JELKESZLET MENTESE
210 CA%=18765 :REM TARTALOMJEGYZEK
220 BE%=18689 :REM UTASITAS AZONOSITASA
230 Q  =13048 :REM PROBA JEL CIME
240 REM
250 REM VEZERLOJELEK
260 REM ************
270 C0$=CHR$(147) :REM KEPERNYOTORLES
280 C1$=CHR$( 19) :REM HOME
290 C2$=CHR$(183) :REM FELSO VONAL
300 C3$=CHR$(117)+CHR$( 99)+CHR$(105):REM FELSO SZ.
310 C4$=CHR$(106)+CHR$( 99)+CHR$(107):REM ALSO SZEL
320 C5$=CHR$( 98):REM KOZEPSO FUGGOLEGES VONAL
330 C6$=CHR$( 18):REM RVS ON
340 C7$=CHR$(146):REM RVS OFF
350 NA%=704 :REM FILE-NEV HOSSZA ($02C0)
360 GA%=186 :REM KESZULEKCIM ($BA)
370 MO%=763 :REM UZEMMOD
380 TA%=764 :REM BILLENTYU/UTASITASKOD
390 YK%=765 :REM Y-KOORD.
400 REM
410 REM SZINEK DEFINIALASA
420 REM ******************
430 DATA 144,  5, 28,159,156, 30, 31,158
440 DATA 129,149,150,151,152,153,154,155
450 DIM C$(16)
455 FOR Y=0 TO 15:READ X:C$(Y)=CHR$(X):NEXT Y
460 N=1:F(0)=0:F(1)=1::V$=" ":SYS IN%
500 REM
510 REM TORLES
```

```
520 REM ******
530 FOR Y=Q TO Q+7:POKE Y,0:NEXT Y:REM PROBAJEL TORL.
540 PRINT C0$
550 PRINT C1$;SPC(10);C$(7);"  JEL  TERVEZES"
560 PRINT SPC(11);C$(1);"(C) AXEL PLENGE"
570 PRINT C$(4);:FOR X=1 TO 40:PRINT C2$;:NEXT X
575 PRINT C$(6):PRINT
580 PRINT"    76543210":SYS NE%
590 PRINT"    ";:FOR X=0 TO 7:PRINT C2$;:NEXT X
600 A=15:B=5:GOSUB 9000:REM POZICIONALAS
610 PRINT C3$:PRINT TAB(15);C5$;CHR$(127)
615 POKE 55552,F(1):REM SZINES PROBA-JEL
620 PRINT TAB(15);C4$:GOSUB 3000:X=0:Y=0
625 REM ALLAPOTMEZO X/Y-KOORD.
700 REM
710 REM BEVITELI CIKLUS
720 REM ***************
730 A=X+3:B=Y+6:GOSUB 9000:REM POZICIONALAS
740 POKE XK%,X:POKE YK%,Y:F=0:REM KOORD.-K ATADASA
750 PRINT C$(7);C6$;" ";CHR$(157);:REM VILLOGAS BE
760 FOR S=1 TO 50:GET A$:IF A$<>"" THEN 800
770 NEXT S:SYS PU%
775 FOR S=1 TO 50:GETA$:IFA$="" THEN NEXTS:GOTO750
776 REM KIKAPCSOLAS
800 REM
810 REM UTASITAS AZONOSITAS
820 REM *******************
830 SYS PU%:C=ASC(A$):POKE TA%,C:SYS BE%
835 S=PEEK(TA%):REM UTASITAS ATADAS/VISSZAJELZES
840 REM RESZLETEZES:
850 ON S GOTO 1600,2800,2900,1060,1060,1080,1080,1100,1100
860 ON S-9 GOTO 1110,1110,3100,3100,3100,3100
870 ON S-15 GOTO 3100,3100,3100,3100, 500,1700
880 ON S-21 GOTO 1800,1900,2000,2100,1200,3400
890 ON S-27 GOTO 1300,3200,1400, 700
1000 REM
1010 REM UTASITAS FELDOLGOZAS
1020 REM ********************
1030 REM
1040 REM KURZOR MOZGATAS
1050 REM ***************
1060 X=X+1:IF X=8  THEN X=0:GOTO 1100
1070 GOTO 700:REM JOBBRA
1080 X=X-1:IF X<0 THEN X= 7:GOTO 1110
1090 GOTO 700:REM BALRA
1100 Y=(Y+1) AND 7:GOTO 700:REM LE
1110 Y=(Y-1) AND 7:GOTO 700:REM FEL
1190 GOTO 690
1200 REM
1210 REM BEFEJEZES
1220 REM *********
1230 A=2:B=15:GOSUB 9000:REM POZICIONALAS
1240 PRINT C6$;C$(7);"VEGE (I/N)?";C7$;C$(6)
1250 INPUT T$:IF T$="I" THEN SYS 64738
1260 GOTO 540
```

```
1300 REM
1310 REM TARTALOMJEGYZEK
1320 REM ***************
1330 PRINT C0$:SYS CA%:GOSUB 9100:GOTO 540
1400 REM
1410 REM ELTOLAS
1420 REM *******
1430 GOSUB 8999:PRINT C$(1);"ELTOLAS :";
1440 PRINT "JOBBRA (J),BALRA (B)"
1445 PRINT SPC(9)"FEL     (F),LE     (L)"
1450 GOSUB 9100:IF T$<>"J" THEN 1470
1460 FOR T=0 TO 7:R=PEEK(Q+T)
1465 POKE Q+T,(R/2)+(R AND 1)*128:NEXT T:REM JOBBRA
1470 IF T$<>"B" THEN 1490
1480 FOR T=0 TO 7:R=PEEK(Q+T)
1485 POKE Q+T,(R AND 255)+R/256:NEXT T:REM BALRA
1490 S= 1:IF T$= "F" THEN 1510
1500 S=-1:IF T$<>"L" THEN 1520
1510 FOR T=0 TO 7:P(T)=PEEK((7 AND T+S)+Q):NEXT T
1515 FOR T=0 TO 7:POKE Q+T,P(T):NEXT T:REM FEL/LE
1520 GOSUB 9200:GOTO 550
1600 REM
1620 REM UZEMMOD VALTAS
1630 REM **************
1640 M=ABS(M-1):POKE MO%,M:GOSUB 3000:GOTO 700
1700 REM
1710 REM A JEL DEFINIALASA
1720 REM *****************
1730 GOSUB 8999:PRINT ;C6$;C$(5);"MELYIK JEL HELYETT?";
1740 PRINT C$(7);C$(6);CHR$(127);C$(7)
1750 GOSUB 2500:IF F=1 THEN F=0:GOTO 540
1760 FOR X=0 TO 7:POKE T+X,PEEK(Q+X):NEXT X
1765 REM TAROLAS
1770 FOR X=1 TO 1500:NEXT X:GOSUB 9200:GOTO 550
1775 REM VARAKOZAS+TORLES
1800 REM
1810 REM A JEL SZARMAZASA
1820 REM ****************
1830 GOSUB 8999:PRINT C$(1);"KEREM A FELDOLGOZANDO"
1835 PRINT"JELET:"
1840 GOSUB 2500:IF F=1 THEN F=0:GOTO 540
1850 FOR Y=0 TO 7:POKE Q+Y,PEEK(T+Y):NEXT Y
1855 GOSUB 9200:GOTO 550:REM TOLTES
1900 REM
1910 REM INVERTALAS
1920 REM **********
1930 FOR B=0 TO 7:POKE Q+B,255-PEEK(Q+B)
1935 NEXT B:GOTO 550
2000 REM
2010 REM MENTES
2020 REM ******
2030 GOSUB 8999:PRINT C6$;C$(1);"JELKESZLET"
2035 PRINT C6$;"MENTESE";C7$
2040 GOSUB 2300:IF F=1 THEN F=0:GOTO 2000
2045 REM BEVITEL/HIBAELLENORZES
```

```
2050 IF F=2 THEN F=0: GOTO 2150:REM HIBA
2060 SYS SP%:GOTO 2200:REM MENTES
2100 REM
2110 REM BETOLTES
2120 REM ********
2130 GOSUB 8999:PRINT C6$;C$(1);"JELKESZLET";
2135 PRINT" TOLTESE:";C7$
2140 GOSUB 2300:IF F=1 THEN F=0:GOTO 2100
2150 IF F=2 THEN F=0:GOTO 540
2160 SYS LA%
2200 REM HIBAELLENORZES (CSAK LEMEZRE!)
2210 OPEN 1,8,15:INPUT#1,DS,DS$,DT,DB:CLOSE 1
2220 IF DS<20 THEN 500:REM OK
2230 PRINT
2235 T$=STR$(DS)+","+DS$+","+STR$(DT)+","+STR$(DB)
2240 GOSUB 9310:PRINT T$:FOR S=1 TO 2000:NEXT S
2245 GOTO 500:REM VILLOGAS
2300 REM
2310 REM NEV MEGADASA
2320 REM ************
2330 A$="":PRINT:PRINT"FILE-NEV"+C$(6);
2335 INPUT A$:T=LEN(A$)
2340 S=VAL(RIGHT$(A$,1))
2350 IFS<>0ANDLEFT$(RIGHT$(A$,2),1)=";"THENT=T-2:POKEGA%,S
2360 IF T=0 THEN F=2:RETURN
2370 IF T>17 THEN 2390
2375 REM NEV A GEPI RUTINNAK (704=$02C0)
2380 POKE NA%,T:FOR S=1 TO T
2385 POKE NA%+S,ASC(MID$(A$,S,1)):NEXT S:RETURN
2390 PRINT CHR$(145);:GOSUB 970
2395 T$=C6$+"HOSSZU!"+C7$:GOSUB 915:REM HIBAUZENET
2400 PRINT C$(6):F=1:RETURN
2500 REM
2510 REM CIM KISZAMITASA
2520 REM ***************
2530 PRINT "JELKESZLET(1-4):";N:A$=""
2540 PRINT "BILLENTYU(F3) VAGY ASCII(F5)?"
2545 GOSUB 9100
2550 ON ABS(ASC(T$)-132) GOTO 2570,2570,2590,2590
2560 GOTO 9300
2570 INPUT"BILLENTYU";A$:A$=LEFT$(A$,1)
2575 IF A$="" THEN 9300:REM BEVITEL BILLENTYUZETROL
2580 T=ASC(A$):GOTO 2620
2590 INPUT"ASCII";A$:IF A$="" THEN 9300
2600 T=VAL(A$)/255:T=INT((T-INT(T))*255):A$=CHR$(T)
2610 IF T<32 OR (T<160 AND T>127) THEN 9300
2620 IF T>191 THEN T=T-96:REM ASCII-ATALAKITAS
2630 PRINT CHR$(145);"BILL.:";A$;" ASCII:";T;
2635 IF B>8 THEN V$=A$
2640 REM ATSZAMITAS (T=A JEL NORMAL ASCII-KODJA)
2650 IF T<64 THEN S=256:GOTO 2680
2660 IF T<128 THEN S=256*((32 AND T)/16)
2670 IF T>159 THEN S=256*(INT(T/32)-2)
2680 T=(N+47)*1024+S+(31 AND T)*8:RETURN
2800 REM
```

```
2810 REM JELKESZLET VALTAS
2820 REM *****************
2830 A=36:B=5:GOSUB 9000:PRINT C$(2);C6$;N;C7$;
2835 PRINT C$(6):REM SZAM INVERTALASA
2840 GOSUB 9100:S=VAL(T$)
2845 IF S=0 OR S>4 THEN S=N:REM S=UJ/N=REGI JELK.
2850 N=S:GOSUB 3000:GOTO 700
2900 REM
2910 REM ?????(1-ES UZEMMOD)
2920 REM ********************
2930 IF C<32 OR (C>127 AND C<160) THEN 700
2940 T=C:R=N:V$=A$:GOSUB 3000:GOTO 1850
3000 REM
3010 REM ALLAPOTMEZO ELOALLITASA
3020 REM ***********************
3030 A=20:B=5:GOSUB 9000
3035 PRINT C$(7);"UZMOD:";M;"/JELK.:";N:A$=V$:T=ASC(A$)
3040 PRINT TAB(20);CHR$(17);CHR$(17);C$(6);
3050 GOSUB 2620:PRINT CHR$(157);" ":PRINT
3060 PRINT TAB(20);C$(7);"SZINEK:"
3070 PRINT TAB(20);C$(2);:FOR S=1 TO 14
3080 FOR S=0 TO 1:PRINT TAB(20);"ALAPSZ.";S;":";F(S)
3085 NEXT S:RETURN
3100 REM
3110 REM RAJZOLAS
3120 REM ********
3130 S=((S-12) AND 2)/2:REM ???
3140 T=2↑(7-X):POKE Q+Y,PEEK(Q+Y) AND (255-T)OR S*T
3145 GOTO 700
3200 REM
3210 REM SZINSKALA
3220 REM *********
3230 GOSUB 8999
3235 PRINT TAB(4);C$(1)"S"C$(2)"Z"C$(3)"I";
3240 PRINT C$(4)"N"C$(5)"S"C$(6)"K"C$(7)"A";
3245 PRINT C$(4)"L"C$(6)"A"C$(2)" "C$(7)":"
3250 PRINT TAB(4);C$(1);CHR$(172);:FOR S=1 TO 32
3255 PRINT CHR$(162);:NEXT S:PRINT CHR$(187)
3260 FOR S=1 TO 2:PRINT TAB(4);C6$;CHR$(161);
3270 FOR T=0 TO 15:PRINT C$(T);"  ";:NEXT T
3275 PRINT C$(1);C7$;CHR$(161):NEXT S
3280 PRINT TAB(4);C$(6);CHR$(182);
3285 PRINT" 0 1 2 3 4 5 6 7 8 9101112131415";
3286 PRINT C7$;CHR$(161)
3290 PRINT:PRINT C$(6);"    ALAPSZINEK (F1/F3): ";
3300 GOSUB 9100:T=ASC(T$)-133:REM FUNKCIOBILL.-K
3310 IF T<0 OR T>1 THEN GOSUB 9300:GOTO 540
3315 REM HIBA
3320 IF T>1 THEN T=T-4
3330 PRINT T:T$="":INPUT"    SZIN  ";T$
3335 S=ABS(INT(VAL(T$)))
3340 IF T$="" OR S>15 THEN GOSUB 9300:GOTO 540
3345 REM HIBA
3350 F(T)=S:POKE 53281+T,S:REM SZINEK BEALLITASA
3360 GOTO 540
```

```
3400 REM
3410 REM UTASITASKESZLET
3420 REM ***************
3430 PRINT C0$;C6$;C$(2)"                ";C$(7);
3440 PRINT" UTASITASKESZLET";C$(2);"                ";
3445 PRINT C7$;
3450 PRINT C$(4);:FOR S=1 TO 40:PRINT CHR$(184);
3455 NEXT S:PRINT
3460 PRINT C$(1)"SZAMA "C6$" UTASITAS "C7$" -";
3465 PRINT C$(5)" FELADATA"C$(4)
3470 FOR S=1 TO 10:PRINT"----";:NEXT S
3480 PRINT C$(1)" (1)  "C6$"(<>↑..)   "C7$" -";
3485 PRINT C$(5)" KURZOR MOZGATAS        "
3490 PRINT C$(1)" (2)  "C6$"(<DEL>)   "C7$" -";
3495 PRINT C$(5)" UZEMMOD VALTAS         "
3500 PRINT C$(1)" (3)  "C6$"(CTRL↑)   "C7$" -";
3505 PRINT C$(5)" JELKESZLET VALTAS      "
3510 PRINT C$(1)" (4)  "C6$"(F1-F8)   "C7$" -";
3515 PRINT C$(5)" RAJZ A 0-15 SZINNEL "
3520 PRINT C$(1)" (5)  "C6$"(F)       "C7$" -";
3525 PRINT C$(5)" F1/F3 SZINEK           "
3530 PRINT C$(1)" (6)  "C6$"(B)       "C7$" -";
3535 PRINT C$(5)" UTASITASKESZLET        "
3540 PRINT C$(1)" (7)  "C6$"(D)       "C7$" -";
3545 PRINT C$(5)" JEL DEFINIALASA        "
3550 PRINT C$(1)" (8)  "C6$"(H)       "C7$" -";
3555 PRINT C$(5)" JEL SZARMAZASA         "
3560 PRINT C$(1)" (9)  "C6$"(I)       "C7$" -";
3565 PRINT C$(5)" A JEL INVERTALASA      "
3570 PRINT C$(1)"(10)  "C6$"(V)       "C7$" -";
3575 PRINT C$(5)" A JEL ELTOLASA         "
3580 PRINT C$(1)"(11)  "C6$"(L)       "C7$" -";
3585 PRINT C$(5)" A JEL TORLESE          "
3590 PRINT C$(1)"(12)  "C6$"(CTRL/G)  "C7$" -";
3595 PRINT C$(5)" A JEL BETOLTESE        "
3600 PRINT C$(1)"(13)  "C6$"(CTRL/S)  "C7$" -";
3605 PRINT C$(5)" A JEL MENTESE          "
3610 PRINT C$(1)"(14)  "C6$"(C)       "C7$" -";
3615 PRINT C$(5)" TARTALOMJEGYZEK        "
3620 PRINT C$(1)"(15)  "C6$"(CTRL/X)  "C7$" -";
3625 PRINT C$(5)" BEFEJEZES              "
3630 GOSUB 9100:GOTO 540
8000 REM
8100 REM ALPROGRAMOK
8200 REM ***********
8300 REM
8400 REM POZICIONALAS
8500 REM ************
8999 A=0:B=16:REM UZENET-MEZO
9000 PRINT C1$;:FOR S=2 TO B:PRINT:NEXT S
9010 PRINT TAB(A);:RETURN
9050 REM
9060 REM BEVITEL A BILLENTYUZETROL
9070 REM *************************
9100 POKE198,0:WAIT 198,255:GET T$:RETURN
```

```
9150 REM
9160 REM UZENET-MEZO TORLESE
9170 REM *******************
9200 A=0:B=16:GOSUB 9000:FOR S=1 TO 5:GOSUB 9210
9205 PRINT:NEXT S:RETURN
9210 FOR T=1 TO 9:PRINT"    ";:NEXT T:RETURN
9250 REM
9260 REM HIBAUZENET (VILLOGAS)
9270 REM *********************
9300 T$="NEM MEGENGEDETT !"
9310 PRINT C$(1):FOR S=1 TO 9:PRINT T$
9315 GOSUB 9330:PRINT CHR$(145);
9320 GOSUB 9210:PRINT CHR$(145)
9325 GOSUB 9330:NEXT S:GOSUB 9200:F=1:RETURN
9330 FOR T=1 TO 75:NEXT T:RETURN:REM VARAKOZAS
9890 REM
9900 REM **********************
9910 REM **                  **
9920 REM ** GEPI KODU RUTINOK **
9930 REM **                  **
9940 REM **********************
9950 REM
9960 REM INDITASKOR AZ ADATOK TORLODNEK !!
9970 REM
10000 FOR I=1 TO 16:READ X:NEXT I
10002 REM AZ ELSO 16 ADAT ATUGRASA (SZINEK)
10005 FOR I=18432 TO 18836
10010 READ X:POKE I,X:S=S+X:NEXT I
10020 DATA 120,169, 51,133,  1,169
10025 DATA  48, 32, 26, 72,169,192
10030 DATA  32, 26, 72,169, 55,133
10035 DATA   1,169, 28,141, 24,208
10040 DATA  88, 96,133,  5,169,208
10045 DATA 160,  0,132,  2,132,  4
10050 DATA 133,  3,162, 16,177,  2
10055 DATA 145,  4,136,208,249,230
10060 DATA   5,230,  3,202,208,242
10065 DATA  96,173,192,  2,162,193
10070 DATA 160,  2, 32,249,253,169
10075 DATA   2,166,186,160,  0, 32
10080 DATA   0,254,169,  0,162,  0
10085 DATA 160,192, 76,213,255,173
10090 DATA 192,  2,162,193,160,  2
10095 DATA  32,249,253,169,  2,166
10100 DATA 186,160,  0, 32,  0,254
10105 DATA 169, 20,141, 24,208,169
10110 DATA  48,133,  5,169,192, 32
10115 DATA  30, 72,169,  0,133,  2
10120 DATA 169, 48,133,  3,169,  2
10125 DATA 162,255,160, 63, 32,216
10130 DATA 255,120,169, 48,162, 51
10135 DATA 134,  1, 32, 26, 72,169
10140 DATA  55,133,  1,169, 28,141
10145 DATA  24,208, 88, 96,160,  0
10150 DATA 169, 32, 32,210,255, 32
```

```
10155 DATA 210,255,152,  9, 48, 32
10160 DATA 210,255,162,  0, 32,207
10165 DATA  72,169, 29, 32,210,255
10170 DATA 232,224,  8,208,243,169
10175 DATA 165, 32,210,255,169, 13
10180 DATA  32,210,255,200,192,  8
10185 DATA 208,212, 96,174,254,  2
10190 DATA 172,253,  2,152, 72,138
10195 DATA  72,169,  0, 56,106,202
10200 DATA  16,252, 57,248, 50,208
10205 DATA   3,160,  0, 44,160,  6
10210 DATA 162,  6,185,245, 72, 32
10215 DATA 210,255,200,202,208,246
10220 DATA 104,170,104,168, 96,146
10225 DATA  31,111, 31,146,157, 18
10230 DATA  28, 32, 31,146,157,173
10235 DATA 252,  2,162,  0,201, 20
10240 DATA 240, 35,201,148,240, 31
10245 DATA 232,201, 30,240, 26,201
10250 DATA  94,208,  5,172,251,  2
10255 DATA 240, 17,232,172,251,  2
10260 DATA 208, 11,160, 28,232,221
10265 DATA  47, 73,240,  3,136,208
10270 DATA 247,232,142,252,  2, 96
10275 DATA  87, 29, 81,157, 65, 17
10280 DATA  50,145,133,137,134,138
10285 DATA 135,139,136,140, 76, 68
10290 DATA  72, 73, 19,  7, 24, 66
10295 DATA  67, 70, 86,169, 36,133
10300 DATA   2,169,  1,162,  2,160
10305 DATA   0, 32,249,253,169,  2
10310 DATA 166,186,160,  0, 32,  0
10315 DATA 254,169,  0,162,  0,160
10320 DATA  64,134, 95,132, 96, 32
10325 DATA 213,255,165, 95,164, 96
10330 DATA  32, 55,165,173,  0,  3
10335 DATA  72,173,  1,  3, 72,169
10340 DATA  61,141,  0,  3,169,227
10345 DATA 141,  1,  3, 32,195,166
10350 DATA 104,141,  1,  3,104,141
10355 DATA   0,  3, 96
10360 IF S=46617 THEN PRINT"RENDBEN !":RETURN
10370 PRINT"HIBA A DATA-SOROKBAN !":END
```

```
                          ;*** P 31 ***
                          ;
                          ;
                          ;
                          ;GEPI RUTIN
                          ;**********
                          ;
                          ;
                          ;
                          ;
4800                              *=      $4800     ;
                          ;
                          ;UGRASI CIMEK ES REGISZTEREK
                          ;***************************
                          ;
4800                      CHROUT   =      $FFD2     ;JELKIVITEL
4800                      FNPAR    =      $FDF9     ;FILENEV PARAMETER
4800                      FPAR     =      $FE00     ;FILEPARAMETER
4800                      SAVE     =      $FFD8     ;MENTES LEMEZ/KAZETTA
4800                      LOAD     =      $FFD5     ;BETOLTES LEMEZ/KAZETTA
4800                      MINTA    =      $32F8     ;MINTAJEL
4800                      HOSSZ    =      $02C0     ;FILENEV HOSSZA
4800                      UZEMM    =      $02FB     ;UZEMMOD
4800                      BILL     =      $02FC     ;UTASITASBILLENTYU
4800                      YKOORD   =      $02FD     ;MEZO-Y-KOORDINATA
4800                      XKOORD   =      $02FE     ;MEZO-X-KOORDINATA
                          ;
                          ;JELKESZLET MASOLASA
                          ;*******************
                          ;
4800 78                   INIC     SEI              ;MEGSZAKITAS TILTASA
4801 A9 33                         LDA    #$33
4803 85 01                         STA    $01       ;A JELKESZLET OLVASHATO
4805 A9 30                         LDA    #$30      ;$D000-TOL $3000-IG
4807 20 1A 48                      JSR    MOVE      ;MASOLAS
480A A9 C0                         LDA    #$C0      ;VON $D000 NACH $C000
480C 20 1A 48                      JSR    MOVE      ;MASOLAS
480F A9 37                         LDA    #$37
4811 85 01                         STA    $01       ;I/O KIVALASZTASA
4813 A9 1C                         LDA    #$1C
4815 8D 18 D0                      STA    $D018     ;JELKESZLET CIME $3000-RE
4818 58                            CLI              ;MEGSZAKITAS ENGEDELYEZESE
4819 60                            RTS
                          ;
                          ;MASOLAS
                          ;*******
                          ;
481A 85 05                MOVE     STA    $05       ;CELCIM FELSO BYTE
481C A9 D0                         LDA    #$D0      ;FORRASCIM FELSO BYTE
481E A0 00                M0       LDY    #$00
4820 84 02                         STY    $02       ;FORRASCIM ALSO BYTE
4822 84 04                         STY    $04       ;CELCIM ALSO BYTE
4824 85 03                         STA    $03
4826 A2 10                         LDX    #$10
4828 B1 02                M1       LDA    ($02),Y   ;BETOLTES
```

```
4828 91 04               STA  ($04),Y ;MENTES
482C 88                  DEY
482D D0 F9               BNE  M1
482F E6 05               INC  $05
4831 E6 03               INC  $03
4833 CA                  DEX
4834 D0 F2               BNE  M1        ;KOVETKEZO OLDAL
4836 60                  RTS
                         ;
                         ;JELKESZLET TOLTESE
                         ;******************
                         ;
4837 AD C0 02 TOLT       LDA  HOSSZ     ;NEV HOSSZA
483A A2 C1               LDX  #$C1      ;FILENEV CIM ALSO BYTE
483C A0 02               LDY  #$02      ;FELSO BYTE
483E 20 F9 FD            JSR  FNPAR
4841 A9 02               LDA  #$02      ;LOGIKAI FILESZAM
4843 A6 BA               LDX  $BA       ;KESZULEKCIM
4845 A0 00               LDY  #$00      ;SZEKUNDERCIM
4847 20 00 FE            JSR  FPAR
484A A9 00               LDA  #$00      ;LOAD/VERIFY-JELZO
484C A2 00               LDX  #$00      ;KEZDOCIM(ALSO BYTE)
484E A0 C0               LDY  #$C0      ;KEZDOCIM(FELSO BYTE)
4850 4C D5 FF            JMP  LOAD
                         ;
                         ;JELKESZLET TAROLASA
                         ;*******************
                         ;
4853 AD C0 02 MENTES     LDA  HOSSZ     ;FILENEV HOSSZA
4856 A2 C1               LDX  #$C1      ;FILENEV CIME(ALSO)
4858 A0 02               LDY  #$02      ;FILENEV CIME(FELSO)
485A 20 F9 FD            JSR  FNPAR
485D A9 02               LDA  #$02      ;LOGIKAI FILESZAM
485F A6 BA               LDX  $BA       ;KESZULEKCIM
4861 A0 00               LDY  #$00      ;SZEKUNDERCIM
4863 20 00 FE            JSR  FPAR
4866 A9 14               LDA  #$14
4868 8D 18 D0            STA  $D018     ;JELKESZLET EREDETI HOSSZA
486B A9 30               LDA  #$30      ;CELCIM(FELSO BYTE)
486D 85 05               STA  $05
486F A9 C0               LDA  #$C0      ;FORRASCIM(FELSO BYTE)
4871 20 1E 48            JSR  M0        ;$C000-$3000
4874 A9 00               LDA  #$00
4876 85 02               STA  $02       ;KEZDOCIM(ALSO BYTE)
4878 A9 30               LDA  #$30
487A 85 03               STA  $03       ;KEZDOCIM(FELSO BYTE)
487C A9 02               LDA  #$02      ;KEZDOCIM NULLASLAP CIMEI
487E A2 FF               LDX  #$FF      ;VEGCIM(ALSO BYTE)
4880 A0 3F               LDY  #$3F      ;VEGCIM(FELSO BYTE)
4882 20 D8 FF            JSR  SAVE      ;ZS TAROLASA $3000-TOL
4885 78                  SEI            ;MEGSZAKITAS TILTASA
4886 A9 30               LDA  #$30      ;CELCIM(FELSO BYTE)
4888 A2 33               LDX  #$33
488A 86 01               STX  $01       ;JELKESZLET OLVASHATO
488C 20 1A 48            JSR  MOVE      ;$D000-$3000
```

```
488F A9 37              LDA  #$37
4891 85 01              STA  $01      ;I/O KIVALASZTASA
4893 A9 1C              LDA  #$1C
4895 8D 18 D0           STA  $D018    ;JELKESZLET CIME $3000-RE
4898 58                 CLI           ;MEGSZAKITAS ENGEDELYEZESE
4899 60                 RTS           ;VISSZA A BASICBE
                  ;
                  ;MUNKAMEZO LETREHOZASA
                  ;**********************
                  ;
489A A0 00     HALO     LDY  #$00     ;SORSZAMALALO=0
489C A9 20     N0       LDA  #$20     ;" "
489E 20 D2 FF           JSR  CHROUT
48A1 20 D2 FF           JSR  CHROUT   ;2 SZOKOZ
48A4 98                 TYA
48A5 09 30              ORA  #$30     ;ZEILENZAEHLER IN ZIFFER WANDELN
48A7 20 D2 FF           JSR  CHROUT   ;KIVITEL
48AA A2 00              LDX  #$00
48AC 20 CF 48 N1        JSR  PONT     ;EGY PONT RAJZOLASA
48AF A9 1D              LDA  #$1D     ;KURZOR JOBBRA
48B1 20 D2 FF           JSR  CHROUT
48B4 E8                 INX           ;KOVETKEZO PONT
48B5 E0 08              CPX  #$08     ;8 PONT/SOR
48B7 D0 F3              BNE  N1
48B9 A9 A5              LDA  #$A5     ;VONAL (CHR$(165))
48BB 20 D2 FF           JSR  CHROUT
48BE A9 0D              LDA  #$0D     ;CARRIGE RETURN
48C0 20 D2 FF           JSR  CHROUT
48C3 C8                 INY           ;KOVETKEZO SOR
48C4 C0 08              CPY  #$08     ;SOR
48C6 D0 D4              BNE  N0
48C8 60                 RTS
                  ;
                  ;EGY KOORDINATA RAJZOLASA
                  ;************************
                  ;
48C9 AE FE 02 PONT2     LDX  XKOORD   ;VEZERLES BASICBOL
48CC AC FD 02           LDY  YKOORD   ;X/Y-KOORDINATA
48CF 98        PONT     TYA
48D0 48                 PHA
48D1 8A                 TXA
48D2 48                 PHA           ;KOORDINATAK MENTESE
48D3 A9 00              LDA  #$00
48D5 38                 SEC           ;EGY BIT BEKAPCSOLASA
48D6 6A        P0       ROR  A        ;A MEGFELELO BIT KERESESE
48D7 CA                 DEX
48D8 10 FC              BPL  P0
48DA 39 F8 32           AND  MINTA,Y ;A TOBBI BIT TORLESE
48DD D0 03              BNE  P1       ;BIT (=PONT) BE "?"
48DF A0 00              LDY  #$00     ;NEM
48E1 2C                 .BYTE$2C      ;BIT-UTASITAS
48E2 A0 06     P1       LDY  #$06     ;BIT BEKAPCSOLASA!
48E4 A2 06              LDX  #$06
48E6 B9 F5 48 P2        LDA  PKTTAB,Y ;JEL A TABLAZATBOL
48E9 20 D2 FF           JSR  CHROUT
```

```
1630:   48EC C8                 INY
1640:   48ED CA                 DEX
1650:   48EE D0 F6              BNE   P2        ;KOVETKEZO JEL
1660:   48F0 68                 PLA
1670:   48F1 AA                 TAX
1680:   48F2 68                 PLA
1690:   48F3 A8                 TAY             ;KOORDINATA VISSZA
1700:   48F4 60                 RTS
                         ;
                         ;JEL EGY KOORDINATAHOZ
                         ;********************
                         ;
1750:   48F5 92 1F 6F PKTTAB    .BYTE146,031,111,031,146,157
1760:   48FB 12 1C 20           .BYTE018,028,032,031,146,157
                         ;
                         ;UTASITASBILLENTYUK AZONOSITASA
                         ;******************************
                         ;
1810:   4901 AD FC 02 UTAZ      LDA   BILL      ;BILLENTYUKOD
1820:   4904 A2 00              LDX   #$00      ;UTASITASKOD
1830:   4906 C9 14              CMP   #$14      ;DEL (=UZEMMOD VALTAS)"?"
1840:   4908 F0 23              BEQ   B2        ;IGEN
1850:   490A C9 94              CMP   #$94      ;INZERT (MINT DEL)"?"
1860:   490C F0 1F              BEQ   B2        ;IGEN
1870:   490E E8                 INX             ;UTASITASKOD+1
1880:   490F C9 1E              CMP   #$1E      ;CTRL"↑""?"
1890:   4911 F0 1A              BEQ   B2        ;IGEN
1900:   4913 C9 5E              CMP   #$5E      ;"↑""?"
1910:   4915 D0 05              BNE   B0        ;NEM
1920:   4917 AC FB 02           LDY   UZEMM     ;CSAK 0-AS UZEMMOD
1930:   491A F0 11              BEQ   B2
1940:   491C E8           B0    INX             ;UTASITASKOD+1
1950:   491D AC FB 02           LDY   UZEMM     ;UZEMMOD 1"?"
1960:   4920 D0 0B              BNE   B2        ;IGEN UTAN NINCS TOBB UTASITAS
1970:   4922 A0 1C              LDY   #$1C      ;27-1 UTASITAS (SZAMLALO)
1980:   4924 E8           B1    INX             ;UTASITASKOD NOVELESE
1990:   4925 DD 2F 49           CMP   UTTAB-3,X ;BILL OSSZEHAS A TABL-TAL
2000:   4928 F0 03              BEQ   B2        ;MEGTALALTA
2010:   492A 88                 DEY             ;KOVETKEZO UTASITAS
2020:   492B D0 F7              BNE   B1        ;MEG NINCS KESZ
2030:   492D E8           B2    INX
2040:   492E 8E FC 02           STX   BILL      ;UTASITASKOD VISSZAJELZES
2050:   4931 60                 RTS             ;VISSZA A BASICBE
                         ;
                         ;UTASITASBILLENTYU TABLAZAT
                         ;*************************
                         ;
                         ;W/CRSR-RE/Q/CRSR-LI/A/CRSR-UN
2110:   4932 57 1D 51 UTTAB     .BYTE087,029,081,157,065,017
                         ;2/CRSR-OB/F1/F2/F3/F4
2130:   4938 32 91 85           .BYTE050,145,133,137,134,130
                         ;F5/F6/F7/F8/L/D
2150:   493E 87 8B 88           .BYTE135,139,136,140,076,068
                         ;H/I/<HOME>/CTRL.G/CTRL.X/B
```

```
2170:  4944 48 49 13             .BYTE072,073,019,007,024,066
                          ;C/F/V
2190:  494A 43 46 56             .BYTE067,070,086
                          ;
                          ;LEMEZ TARTALOM
                          ;**************
                          ;
2240:  494D A9 24      TARTAL    LDA   #$24     ;"$"=FILENEV
2250:  494F 85 02                STA   $02
2260:  4951 A9 01                LDA   #$01
2270:  4953 A2 02                LDX   #$02
2280:  4955 A0 00                LDY   #$00     ;S.O.
2290:  4957 20 F9 FD             JSR   FNPAR
2300:  495A A9 02                LDA   #$02
2310:  495C A6 BA                LDX   $BA      ;KESZULEKCIM
2320:  495E A0 00                LDY   #$00
2330:  4960 20 00 FE             JSR   FPAR
2340:  4963 A9 00                LDA   #$00     ;LOAD/VERIFY-JELZO
2350:  4965 A2 00                LDX   #$00
2360:  4967 A0 40                LDY   #$40     ;KEZDOCIM
2370:  4969 86 5F                STX   $5F
2380:  496B 84 60                STY   $60
2390:  496D 20 D5 FF             JSR   LOAD     ;TARTALOM BETOLTESE PROGRAMKENT
2400:  4970 A5 5F                LDA   $5F      ;SZIMULACIO
2410:  4972 A4 60                LDY   $60      ;BASIC-PROGRAM KEZDOCIME
2420:  4974 20 37 A5             JSR   $A537    ;BASICSOROK KOTESE
2430:  4977 AD 00 03             LDA   $0300
2440:  497A 48                   PHA
2450:  497B AD 01 03             LDA   $0301    ;MELEGINDITAS MUTATOJA
2460:  497E 48                   PHA            ;MENTES
2470:  497F A9 3D                LDA   #$3D
2480:  4981 8D 00 03             STA   $0300
2490:  4984 A9 E3                LDA   #$E3
2500:  4986 8D 01 03             STA   $0301    ;ES RTS
2510:  4989 20 C3 A6             JSR   $A6C3    ;LIST UTASITAS VEGREH
2520:  498C 68                   PLA
2530:  498D 8D 01 03             STA   $0301
2540:  4990 68                   PLA
2550:  4991 8D 00 03             STA   $0300    ;REGI MUTATO VISSZA
2560:  4994 60                   RTS            ;VISSZA A BASICBA
```

Ez a jeltervező program a sprite-tervezővel azonos elven működik. Itt is kiválaszthatjuk a színeket, használhatjuk a másodlagos műveleteket, továbbá a képernyőn rendelkezésünkre áll egy 8×8-as munkamező és egy eredeti méretű mintamező. Ehhez jön még az ún. üzemmód és a készlet kiválasztása.

A program kétféle üzemmódban dolgozik:

0: Írás üzemmód

Ebben az üzemmódban minden utasítás használható és a jelek megszerkeszthetők.

1: Betöltő üzemmód

Ebben az üzemmódban a billentyűzeten keresztül lehívhatjuk a jelkészlet aktuális elemeit, amik a jelkészletpufferba kerülnek. (A program indítása után a jelkészlettárba a teljes eredeti jelkészlet kerül.)

Négyféle jelkészlettípust különböztetünk meg, amelyeket 1–4-ig számozunk. Ezek sorrendben a következők:

1	–	normál	nagybetű/grafikus jelek,
2	–	inverz	nagybetű/grafikus jelek,
3	–	normál	kis-/nagybetű,
4	–	inverz	kis-/nagybetű.

Mindig az az aktuális, amelyiket a jelkészlet kiválasztásakor megneveztünk. A sprite-tervező program utasításai – a G-t kivéve – ennél a programnál is használhatók, kiegészítve a következőkkel:

DEL-billentyű

Üzemmódváltás (0–1).

CTRL/↑

Ha jelkészletet akarunk választani, a két billentyű lenyomása után adjuk be a megfelelő számot (1–4). Írás üzemmódban a ↑-billentyű lenyomása elegendő.

D/H

D: az új jelhez ASCII-kódot vagy billentyűt rendelhetünk és ezzel a jelkészlettárba írhatjuk.

H: a már létező jelet a jelkészletpufferból a munkamezőbe írhatjuk (mint az 1-es üzemmódban).

Mindkét esetben az aktuális jelkészlet működik. El kell döntenünk, hogy a jeleket ASCII-kódként (F5/F6), vagy billentyűvel (F1/F3) akarjuk-e megadni. Mindkét esetben a RETURN-t is alkalmazni kell.

CTRL/S és CTRL/G

Az új jelkészlet tárolása és betöltése. Ugyanaz, mint a sprite-tervező programnál.

Ha a programmal sikerült létrehoznunk a jelkészletünket, mágneslemezre rögzíthetjük és bármikor egy LOAD"név",8,1 paranccsal betölthetjük a $3000 (12288) címre. Ha máshol szeretnénk elhelyezni, akkor szükségünk van egy monitorprogramra, vagy egy gépi kódú betöltőre. Ezután a következő utasítással át kell állítani a jelkészletmutatót, hogy minderről a VIC is tudomást szerezzen:

```
POKE 53248+24, PEEK (53248) AND 241 OR 12
```

Már említettük, hogy a 24-es VIC-regiszter 1–3 bitjei adják a jelkészlettár 11–13-as címbitjeit, ami a jelkészlettár elhelyezkedését határozza meg. A fenti utasítássorral először töröljük ezt a három bitet (241=%1111 0001), majd a felső kettőt bekapcsoljuk (12=%0000 1100). Ezzel máris megjelenik a képernyőn az új jelkészlet. Ilyen egyszerű az egész!

4.5 A grafika tárolása és betöltése

Sok fáradságba kerül, míg egy grafikus képet elkészítünk. Ha ezt minden alkalommal meg kellene ismételnünk, bizony hamar elmenne a kedvünk a grafikus munkától. Szerencsére létezik néhány eljárás, amivel ez a folyamat jelentősen egyszerűsíthető. Az elkészített képet ugyanis minden további nélkül mágneslemezre vagy kazettára rögzíthetjük, és szükség esetén a tárba tölthetjük. Készíthetünk ún. hardcopyt is, ha megfelelő, pl. egyedi tűvezérléses nyomtatóval rendelkezünk. Így a kép a papíron is rendelkezésünkre áll. Ebben a fejezetben ezek technikai megvalósításáról lesz szó.

Szinte minden nyomtató alkalmas hardcopy készítésére, de az eljárás szinte gépenként különböző. Így sajnos mindegyikhez egyedi hardcopy-rutint kell írni, és valószínűleg az általunk közölt sem fog azonnal működni. Próbáljuk meg átírni saját nyomtatónkra. Szükség esetén a szakirodalomban találunk segítséget. Ha végképp nem boldogulunk vele, vegyünk elő egyet a mi nyomtatónkra is alkalmazható grafikus bővítők közül.

4.5.1 Tárolás és betöltés

A grafika vagy a jelkészlet lemezre vagy kazettára való tárolásánál az a legfőbb probléma, hogy a számítógép számára meg kell adnunk a tárhely kezdő és végcímét, amire viszont nincs megfelelő utasításunk. BASIC-program

esetén ezeket a gép eleve "tudja", de gépi kódú programok és egyéb adatok rögzítésénél a helyzet bonyolultabb.

Bemutatunk egy rövid kis programot, amivel elvileg ez a probléma megoldható. Úgy szerkesztettük meg, hogy a könyvben levő grafikai programok bármelyikébe beépíthető legyen:

```
1 REM *** P 32 ***
2 REM
10 REM *********************
20 REM **                 **
30 REM ** GRAFIKA TAROLASA **
40 REM **                 **
50 REM *********************
60 REM
70 FI$="GRAFIKA" : GA=8:REM FILE-NEV / KESZULEKCIM
80 BE=8192: EN=16192:REM KEZDO- ES VEGCIM
90 GOSUB 11200:END:REM TAROLAS
11200 REM
11210 REM *************
11220 REM ** TAROLAS **
11230 REM *************
11240 REM
11250 SYS 57812 FI$,GA
11255 REM LEMEZEGYSEG PARAMETEREINEK ATADASA
11260 X=EN/256
11270 POKE 175,INT(X):REM VEGCIM FELSO BYT   ($AF)
11280 POKE 174,(X-INT(X))*256:REM ALSO BYTE  ($AE)
11290 X=BE/256
11300 POKE 194,INT(X):REM KEZDOCIM FELSO BYTE($C2)
11310 POKE 193,(X-INT(X))*256:REM ALSO BYTE  ($C1)
11320 SYS62954:REM SAVE-RUTIN ($F5EA)
11330 RETURN
```

A 70–80-as sorokban adjuk meg a file létrehozásához szükséges információkat. A 11250-es sor lehívja a normál BASIC SAVE-utasítás egy részét, ami átveszi a file nevét és a készülékcímet (mágneslemez = 8, kazetta = 1). Ez az utasítás szokatlannak tűnik, de helyes, mert miután SYS57812-vel a SAVE-rutint lehívtuk, erre a két paraméterre lesz szükségünk. Emiatt az írásmód nem eredményez hibaüzenetet.

A következő sorok megadják a 11320-as sorban lehívott SAVE-rutin számára a tárolandó tartomány kezdő és végcímét.

Ez a program jelkészlet, sprite-ok, grafika, szöveg vagy színek mágneslemezre vagy kazettára rögzítésére egyaránt alkalmas. Csupán a paraméterek megválasztásában van különbség.

Fenti példánk a $2000–$3F3F (8192–16192) tártartományban levő grafikát rögzíti. Ha a $3000–$3FFF (12288–16383) tartományban található jelkészletünket akarjuk rögzíteni, a 80-as sort a következők szerint kell módosítanunk:

80 BE = 12288 : EN = 16384

A végcímhez mindig hozzá kell adnunk 1-et.

Egy egész szövegoldal rögzítéséhez a 80-as sor így alakul:

80 BE = 1024 : EN = 2024

A $0400–$07E7 (1024–2024) tártartományban normál esetben a szöveget tároljuk. Grafikus üzemmódban azonban a grafika színtárjaként üzemel és tartalmát ugyanúgy kell tárolni.

Tételezzük fel, hogy lemezre akarjuk rögzíteni a 11-es blokkban levő spritemintát. A 11-es blokk a $02C0–$02FD (704–766) tartományban helyezkedik el, tehát ezeket a címeket kell megadnunk:

80 BE = 704 : EN = 767

Ezzel a módszerrel akármelyik gépi nyelvű programot monitor nélkül is tárolhatjuk.

A tárolt adatok betöltése egyszerűen a

LOAD"név",8,1 (mágneslemez esetén),

vagy a

LOAD"név",1,1 (kazetta esetén)

parancsokkal végezzük.

4.5.2 Hardcopy

A forgalomban levő programok leggyengébb pontja a nyomtató üzemmód. Mivel rengeteg különböző típusú nyomtató létezik, amelyek mindegyike egyedi vezérléssel, ASCII-kóddal és vezérlőjelekkel rendelkezik (különösen a grafika tekintetében), a kereskedelemben kapható programok legtöbbje csak

1–2 típusra alkalmazható. Ezért, ha grafikus bővítőprogramot vagy egyéb más grafikus programot vásárolunk, feltétlenül győződjünk meg arról, hogy használható-e az a rendelkezésünkre álló nyomtatón. Ha még nincs nyomtatónk, olyan programokat érdemes venni, ami többféle típust is vezérelni tud.

Nyomtatóra előbb-utóbb mindenkinek szüksége lesz. Egy hosszabb program írásakor nyomtatott lista nélkül lehetetlen boldogulni. Grafikus munkánál döntő fontosságú a nyomtatás tisztasága, vagyis a nyomtatott szöveg, ill. grafika minősége. Ha választhatunk, ezt is vegyük figyelembe.

Mint már említettük, lehetetlen minden nyomtatótípusra alkalmazható programot bemutatni, ezért itt a SEIKOSHA GP 100VC típussal foglalkozunk, ami 7 tűfejes kiképzésével a legbonyolultabbak közé tartozik. A többi nyomtatóra ebből viszonylag könnyen levezethetők a rutinok. A példában feltételezzük, hogy a grafikustár a \$2000–\$3F3F (8192–16191) tártartományban helyezkedik el.

```
1 REM *** P 33 ***
2 REM
11800 REM ***************************
11810 REM **                       **
11820 REM ** HARDCOPY - GP 100 VC **
11830 REM **      (GRAFIKA)        **
11840 REM ***************************
11850 REM
11860 SA=8192
11870 OPEN1,4:REM NYOMTATOCSATORNAMEGNYITASA
11880 Y=35:MK=255:SO=28:ON=6
11885 REM Y-KOORD./MASZK/SOROK SZAMA/OSZLOP NAGYS.
11890 FOR FLAG=1 TO 2
11900 FOR ZE=1 TO SO:REM SO SOR
11910 PRINT#1,CHR$(8)CHR$(27)CHR$(16)CHR$(0)CHR$(80);
11915 REM KOZPONTOZAS+GRAFIKA BE
11920 XK=0
11930 FOR OS=1 TO 40:YK=Y:REM OSZLOPSZAMLALO
11940 FOR X=0 TO ON:GOSUB 12150:ZW(X)=PEEK(AD)
11945 YK=YK+1:NEXT X:REM 8*7 PONT BETOLTESE
11950 FOR X=0 TO 7:REM 8 BYTE A NYOMTATORA
11960 MS=2↑(7-X):B2=0
11970 FOR Z=0 TO ON:B1=-2*((ZW(Z) AND MS)>0)
11975 B2=B2 OR (B1↑Z+(Z+B1=0)):NEXT Z
11980 B2=(B2 AND MK) OR 128:PRINT#1,CHR$(B2);
11990 NEXT X     :REM KOVETKEZO BYTE A NYOMTATORA
12000 XK=XK+8
12010 NEXT OS    :REM KOVETKEZO OSZLOP
12020 PRINT#1    :REM RETURN
12030 Y=Y+7      :REM Y-KOORD.
12040 NEXT ZE    :REM KOVETKEZO SOR
```

```
12050 SO=1         :REM CSAK AZ UTOLSO SOR
12060 MK=16        :REM MASZK (FELSO 4 BIT LEVAGASA)
12070 ON=4         :REM OSZLOP NAGYSAGA
12080 NEXT FLAG:REM MEGEGYSZER AZ UTOLSO SOR
12090 CLOSE 1:END
12100 REM
12110 REM **********************
12120 REM *** PONT SZAMITASA **
12130 REM **********************
12140 REM
12150 AD=SA+320*INT(YK/8)+(YK AND 7)+8*INT(XK/8)
12155 RETURN
```

Ezt a programot is bevehetjük az alprogram-gyűjteményünkbe. A későbbiekben egy GOSUB-bal a főprogram bármelyik részén lehívható. Ha a grafikát a $2000 (SA=8192) kezdő címtől tároltuk, a megfelelő helyen betöltjük és lefuttatjuk a programot. A grafikustár eközben sértetlen marad és a papíron megjelenik annak hű másolata. Ez a rutin természetesen minden kompatíbilis nyomtatón működik.

A BASIC-rutinok általában elég lassúak, ezért legyünk türelemmel, ha valamire várni kell. Ez a program is szépíthető még a Függelék útmutatása szerint, mert itt az érthetőség kedvéért nem a legegyszerűbb módon jártunk el. Fontos tudnunk, hogy a nyomtató számára 7 egymás alatti pont információit egy byte ad át. A legfelső bit értéke mindig 1, amit a nyomtató igényel. Szokás szerint egy bekapcsolt bit 1 pontot jelent a papíron. Mivel a papíron 1 sor 7 pontból áll, a grafika végén 4 pontsor felesleges marad, mivel a 200 nem osztható maradék nélkül 7-tel. (Emlékeztetőül: a képernyő Y irányban 200 grafikus pontból áll.)

A rutin fő részében (11940–11990) először beolvasásra kerül a 7 egymás alatti grafikus byte (11940), majd ez oszlopról oszlopra átadódik a nyomtatónak (11950–11990). A dolog további részletezése meghaladná könyvünk kereteit, ezért már csak egy kis apróságra hívjuk fel a figyelmet: a 11970-es sorban levő Z+B1 = 0 művelet −1-et ad, ha Z+B1 = 0, és 0-t ad minden ettől eltérő esetben. A 4=6 tehát 0-t, a 4=4 pedig −1-et eredményez.

Próbáljuk ki! Ez a trükk értékes segítséget jelenthet olyan esetekben, amikor döntést kell hoznunk, de valamilyen okból nem szívesen alkalmaznánk az IF...THEN-, vagy ON...GOTO-utasításokat. A dolog lényege az, hogy a CBM 64-es a 0↑0 kifejezés értékeként 1-et ad, ami hibás eredményhez vezetne.

A későbbiekben ezt a hardcopy-rutint assemblerben is megadjuk.

4.6 Megszakítások

A 3.7 fejezetben megismerkedtünk azokkal a megszakítási lehetőségekkel, amelyek a grafikai felhasználás szempontjából érdekesek lehetnek. Akkor a témára vonatkozó elméleti ismereteket már megbeszéltük, és most, némi grafikai előképzettség birtokában, rátérhetünk könyvünk legnehezebb fejezetére: a megszakítás programozására. Már maga az a tény, hogy a megszakítási folyamat csak gépi nyelven kezelhető, az avatatlanok számára riasztó lehet. Az új programtechnikai eljárásokkal talán még a gyakorlott assembler-programozók mindegyike sincs teljesen tisztában. Azért ne ijedjünk meg, és minden példaprogramot bátran próbáljunk ki.

A megszakítási technikát a rasztersor- és a fényceruza-megszakításon keresztül mutatjuk be. Az itt szerzett tapasztalatokat később saját programjainkban (pl. a sprite-ok ütközésvizsgálatában) hasznosíthatjuk. Mielőtt hozzákezdenénk, olvassuk át még egyszer a 3.7 fejezetet.

Idézzük fel röviden azokat a legfontosabb tényezőket, amiket a VIC általános megszakítás-kezeléséről tanultunk. A megszakítás azt jelenti, hogy a programfutás során bekövetkező, valamilyen előre meghatározott esemény a főprogram futásának megszakítását és egy ún. megszakítási rutin végrehajtását eredményezheti. A megszakításnak négy általunk előírt kiváltó oka lehet: egy meghatározott rasztersor elérése, fényceruza-impulzus, sprite–sprite- és sprite–háttérjel-ütközés. Kiválasztásuk az IMR (Interrupt Mask Register), vagyis a 26-os VIC-regiszterben történik. Amikor valamelyik esemény bekövetkezik, azt az IRR (Interrupt Requiest Register), vagyis a 25-ös VIC-regiszter a megfelelő bit bekapcsolásával jegyzi, továbbá a 7. bit értéke is 1 lesz. Ha a 25-ös és 26-os regiszter azonos bitjei egyidejűleg bekapcsolt állapotban vannak, megszakítás történik és a programfutás a megszakítási rutinnal folytatódik. A megszakítási rutin mutatóját a $0314/$0315 (788/789) tárrekeszek rögzítik. A megszakítást gépi nyelven az ún. megszakításjelző (interruptflag) beállításával akadályozhatjuk meg.

4.6.1 Rasztersor-megszakítás

Foglaljuk össze néhány szóban, amit erről a témáról már tudunk.

A megszakításnak ezt a módját mindig az IMR és IRR regiszterek 0. bitjeivel írjuk elő. A 18-as regiszter, valamint a 17-es regiszter 7. bitje adja az aktuális képernyősor számát, amit a VIC éppen felépít. (Ne felejtsük

el, hogy a raszterkoordináták és a grafikus koordináták nem egyeznek meg!) A két regiszter beírásával megadhatjuk, hogy melyik sor felépítésekor következzék be megszakítás. A megszakítás bekövetkezésekor a programot saját megszakítási rutinjainkra irányíthatjuk. A következő assembler-lista is egy ilyen megszakítási rutint szemléltet:

```
                                ;
                                ;*** P34 ***
                                ;
                                ;
       C800                           *=     $C800    ;KEZDOCIM
       C800                  SZIN1    =      $FB
       C800                  SZIN2    =      $FC
       C800                  FEL      =      $FD
       C800                  LE       =      $FE
       C800                  IRQVECT  =      $0314
       C800                  IRQREGI  =      $EA31
       C800                  RASTER   =      $D012
       C800                  IRR      =      $D019
       C800                  IMR      =      $D01A
       C800                  KERET    =      $D020
       C800                  HATTER   =      $D021
                                ;
                                ;INICIALIZALAS
                                ;************
                                ;
       C800 78               INIC     SEI             ;MEGSZAKITAS TILTASA
       C801 A9 1F                     LDA    #<IRQUJ
       C803 8D 14 03                  STA    IRQVECT
       C806 A9 C8                     LDA    #>IRQUJ
       C808 8D 15 03                  STA    IRQVECT+1 ;IRQ-VECTOR ATIRASA
       C80B A5 FD                     LDA    FEL
       C80D 8D 12 D0                  STA    RASTER   ;1.RASTERSOR MEGHATAROZASA
       C810 AD 11 D0                  LDA    RASTER-1
       C813 29 7F                     AND    #$7F
       C815 8D 11 D0                  STA    RASTER-1 ;FELSO-BIT TORLESE
       C818 A9 81                     LDA    #%10000001 ;MASZK
       C81A 8D 1A D0                  STA    IMR      ;RASTERSOR-IRQ KIVALASZTASA
       C81D 58                        CLI             ;IRQ ENGEDELYEZESE
       C81E 60                        RTS
                                ;
                                ;MEGSZAKITASI RUTIN
                                ;******************
                                ;
       C81F AD 19 D0 IRQUJ            LDA    IRR      ;IRR-REGISZTER
       C822 8D 19 D0                  STA    IRR      ;TORLESE
       C825 30 07                     BMI    IRQRAS   ;RASTERSOR-IRQ"?"
                                ;NORMAL IRQ
       C827 AD 0D DC                  LDA    $DC0D    ;CIA 1-IRR TORLESE
       C82A 58                        CLI             ;MEGSZAKITAS ENGEDELYEZESE
       C82B 4C 31 EA                  JMP    IRQREGI ;VISSZA A REGI IRQ-RA
       C82E AD 12 D0 IRQRAS           LDA    RASTER   ;RASTERPOZICIO
       C831 C5 FE                     CMP    LE       ;ALSO ERTEK
```

```
C833 B0 10                 BCS  MASODIK ;IGEN=>UGRAS
C835 A5 FB                 LDA  SZIN1
C837 8D 20 D0              STA  KERET   ;KERET ES
C83A 8D 21 D0              STA  HATTER  ;HATTERSZIN
C83D A5 FE                 LDA  LE      ;ALSO ERTEK A
C83F 8D 12 D0              STA  RASTER  ;RASTERPOZICIOBA
C842 4C BC FE              JMP  $FEBC   ;BEFEJEZES
C845 A5 FC        MASODIK  LDA  SZIN2
C847 8D 20 D0              STA  KERET   ;KERET ES
C84A 8D 21 D0              STA  HATTER  ;HATTERSZIN
C84D A5 FD                 LDA  FEL     ;FELSO ERTEK A
C84F 8D 12 D0              STA  RASTER  ;RASTERPOZICIOBA
C852 4C BC FE              JMP  $FEBC   ;BEFEJEZES
```

A program betöltő listája:

```
1 REM *** P 34 B ***
2 REM
1000 FOR I=51200 TO 51284
1010 READ X:POKE I,X:S=S+X:NEXT I
1020 DATA 120,169, 31,141, 20,  3
1025 DATA 169,200,141, 21,  3,165
1030 DATA 253,141, 18,208,173, 17
1035 DATA 208, 41,127,141, 17,208
1040 DATA 169,129,141, 26,208, 88
1045 DATA  96,173, 25,208,141, 25
1050 DATA 208, 48,  7,173, 13,220
1055 DATA  88, 76, 49,234,173, 18
1060 DATA 208,197,254,176, 16,165
1065 DATA 251,141, 32,208,141, 33
1070 DATA 208,165,254,141, 18,208
1075 DATA  76,188,254,165,252,141
1080 DATA  32,208,141, 33,208,165
1085 DATA 253,141, 18,208, 76,188
1090 DATA 254
1100 IF S=11288 THEN PRINT"RENDBEN !":GOTO 1120
1110 PRINT"HIBA A DATA-SOROKBAN !":END
1120 F1=7:F2=6:REM 1-ES/2-ES SZIN(VONAL=1-ES SZIN)
1130 FE=60:AL=150:REM FELSO/ALSO HATAR
1140 POKE 251,F1:POKE 252,F2
1145 POKE 253,FE:POKE 254,AL
1150 SYS 51200:REM GEPI KODU RUTIN LEHIVASA
1160 REM
1170 FOR X=1 TO 5000:NEXT X:REM VARAKOZAS
1180 REM
1190 REM MOZGATAS
1200 REM ********
1210 FOR X=40 TO 240:POKE 253,X:POKE 254,X+10
1215 NEXT X
1220 GET A$:IF A$="" THEN 1190:REM BILLENTYU ?
```

Indítás után kék háttérben sárga színnel kirajzolódik a képernyő egyik átlója. Ez a keret és a háttérszín folyamatos átkapcsolásával lehetséges, ami mindig akkor történik, amikor a VIC éppen a kijelölt rasztersorban van. Mielőtt a processzor lehívná a megszakítási rutint, néhány dologra még szükség van, amit az ún. előkészítő vagy inicializáló rutin valósít meg. Először is át kell írni a $0314/$0315 tárrekeszekben a megszakítási rutin mutatóját. Ez eredetileg a ROM-ban levő megszakítási rutinra mutat ($C81F=51231). Az új cím alsó és felső byte-ját a 270–300-as sorokkal írjuk be. Ahhoz, hogy eközben ne jöjjön létre 1/60 másodpercenként a szokásos rendszermegszakítás (billentyűzet-lekérdezéshez... stb.), a SEI-utasítással beállítjuk a megszakításjelzőt, vagyis letiltjuk a megszakítást (260-as sor). Ezután meghatározzuk azt a rasztersort, amelynél az első megszakításnak be kell következnie. Ez a sárga sáv felső széle, amit a nullás lap $FD (253) tárrekeszében a megfelelő érték beírásával jelölünk ki. Az értéket a 310–320-as sorokban a VIC 18-as regiszterébe írjuk. Az első megszakítás tehát akkor következik be, amikor a VIC ezt a sort építi fel. Mivel a felső biteket nem használtuk, a 330–350-es sorokban töröljük őket.

A következő fontos lépés az IMR 0. bitjének bekapcsolásával a rasztersor-megszakítás engedélyezése. Az inicializálás végén a megszakításjelzőt ismét töröljük (CLI), így a megszakítások ismét kezelhetők.

A megszakítások kezelésénél problémát jelent, hogy létezik az ún. időzítő által kiváltott megszakítás is (timer interrupt), ami a számítógép rendes működéséhez elengedhetetlen. Mindazok, akik gépi nyelven programoznak és le tudnak mondani a billentyűzet lekérdezéséről, a kurzor villogtatásáról és a TI$-óráról, azok természetesen szabadon választhatnak, hogy használják-e a normál megszakítási rutint, vagy sem. Ezzel azonban csak azok kísérletezzenek, akik a gép operációs rendszerét megfelelően ismerik. Lényegében azt kell eldöntenünk, hogy mi váltsa ki a megszakítást, mert a rutinunk akkor kerül lehívásra, amikor a CIA1-időzítő lefut és a billentyűzet lekérdezése következne. Ehhez először visszaállítjuk (töröljük) az IRR-t, mert különben a rutin végén mindig új megszakítás következne be. Egy rasztersor-megszakítás azt eredményezi, hogy az IRR 0. és 7. bitje bekapcsolódik (3.7 fejezet). A 460-as sorban ellenőrizzük a 7. bitet. Ha értéke 1, a program elágazik az 510-es sorba, a saját megszakítási rutinba. Ellenkező esetben az eredeti megszakítási rutint hajtja végre, aminek kezdőcíme $EA31 (59953).

Itt megint felmerül egy kis probléma. Minden megszakításkor a visszaugrási cím és a CPU flagregisztere automatikusan a veremtárba kerül és – ami nagyon lényeges – beállítódik a megszakításjelző (interruptflag). Ez utóbbinak az a célja, hogy a megszakítási rutin végrehajtásának idejére

megakadályozza egy újabb megszakítás kiváltását. Tételezzük most fel, hogy a program az eredeti megszakítási rutinra ágazik el és megkezdi a feldolgozását. Eközben bekövetkezne egy rasztersor-megszakítás, de a beállított megszakításjelző ezt megakadályozza. Ilyenkor a számítógép is kényszerhelyzetbe kerül, ami a képernyő hirtelen felvillanásában nyilvánul meg, mert a háttér- és keretszínváltás nem történik meg időben. Ezért az időzítőrutin végrehajtásának idejére engedélyezni kell a rasztersor-megszakítást úgy, hogy szükség esetén a normál megszakítási rutint is megszakítjuk. Ez elég komplikáltnak és veszélyesnek tűnhet, de helyes, és ha jól belegondolunk, még logikus is. A normál megszakítási rutin előtt tehát töröljük a megszakításjelzőt – és minden rendben van? Sajnos nem, mert valamit még elfelejtettünk. Mielőtt ugyanis töröljük a megszakításjelzőt, mindig meg kell szüntetnünk a kiváltó okot, esetünkben az időzítőmegszakítást. A CIA1 megszakítási regisztere a $DC0D (56333) címen található. Törlése egyszerű kiolvasással történik, amit a programban a 480-as sor hajt végre. Ezzel az előkészítést befejeztük, rátérhetünk a megszakítási rutinunkra (510-es sor).

Mivel minden képfelépítéskor kétszer kell színt váltani (2-esről az 1-esre és vissza), először meg kell állapítanunk, hogy melyik fázisban vagyunk. Ez az aktuális rasztersor számának kiolvasásával történik (510-es sor), amit az 520-as sorban összehasonlítunk a sárga sáv alsó szélének értékével. Ha az aktuális sor nagyobb vagy egyenlő ezzel az értékkel, át kell kapcsolni a 2-es színre, azaz ugrás a 600-as sorba (530-as sor). Azt is ellenőrizhettük volna, hogy éppen melyik a háttérszín. Ennek is volna néhány előnye. Itt kell megjegyeznünk azt, hogy az is időbe telik, míg a program a megszakítás után erre a helyre ér. A rasztersugár ezalatt néhány sorral tovább fut, ezért igazából sosem azt a rasztersort kapjuk meg aktuálisként, ahol a megszakítás történt. Ezt könnyen beláthatjuk, ha arra gondolunk, hogy már a rutinunk lehívása előtt a ROM-ban levő utasítások közül is néhányat fel kellett dolgozni és ez szintén időbe telt. Ez az oka annak, hogy a raszterbeosztás látszólag nincs összhangban azzal, amit erről a 3.7 fejezetben elmondtunk. A többi már viszonylag egyszerű: módosítjuk a keret és a háttér színét és a következő rasztermegszakítást a felső, ill. alsó sorértékre állítjuk. Végül a normál rutin szabályos befejezésére ugrunk, mert néhány regisztert vissza kell állítani és végre kell hajtani egy RTI-t (ReTurn from Interrupt).

Mint látjuk, a sávot jellemző információkat a nullás lap négy tárrekeszében rögzítettük:

Cím (hexa	-dec)	Információ
$FB	-251	1-es szín
$FC	-252	2-es szín
$FD	-253	felső szél
$FE	-254	alsó szél

Ha ezeket módosítjuk, megváltozik a sáv színe, mérete, alsó és felső széle. Ez legegyszerűbben a P34B-programmal oldható meg, ami betölti a gépi programot, lefekteti a négy paramétert és elindítja az inicializálást és a megszakítást. Az 1200-as sortól a felhasználásra mutatunk példát.

Próbáljunk kitalálni valamit. Írjunk például egy olyan programot, amivel a képernyő minden karaktere egyenként kiemelhető (a vezérlés a kurzorbillentyűkkel történjen). Esetleg ha már kiismertük magunkat a gépi nyelvű programozásban, a színek helyett váltogathatjuk a szöveg és grafikus üzemmódot. Vagy vigyünk a képernyőre egyszerre 16 sprite-ot. Rengeteg lehetőség van.

4.6.2 A fényceruza

Ezt a fejezetet akkor is nyugodtan átolvashatjuk, ha nincs fényceruzánk. Sok érdekességet találhatunk benne!

Azt, hogy a fényceruza hogy működik és hogyan illeszkedik a számítógépes folyamatokba, már a 3.7.2 fejezetben megbeszéltük. Nézzük át még egyszer a fontosabb dolgokat!

A fényceruza olyan eszköz, amellyel a képernyő kiválasztott pontjára mutatva jelzéseket küldhetünk a számítógépnek. A számítógép ebből meg tudja állapítani a kiválasztott pont helyzetét. X és Y koordinátáit a VIC 19-es és 20-as regiszterei rögzítik, amelyek kiolvashatók. A koordinátabeosztás a rasztersoroknak felel meg (ld. 3.7 fejezet). A fényceruzával is válthatunk ki megszakítást, amit az IRR- és IMR-regiszterek (VIC 25/VIC 26) 3. bitjével választunk ki.

A fényceruzát az 1-es vezérlőportra (controlport 1) csatlakoztatjuk. Bemenete a joystick tűzgomb bemenetének felel meg.

Ennyi ismétlés után térjünk rá a használatára és a programozására.

A fényceruza lekérdezése kétféleképpen történhet. Az egyszerűbb az, amikor a koordinátákat tároló regisztereket egy BASIC-programmal olvassuk ki. A következő példán ezt mutatjuk be. Ahhoz, hogy a program futtatható legyen, összefűztük a P14-es grafikai programmal. A kizárólag fényceruzára vonatkozó rész a program 1–250-es sorai.

```
1 REM *** P 35 ***
2 REM
100 REM ***************
110 REM **            **
120 REM ** FENYCERUZA **
130 REM **            **
140 REM ***************
150 REM
160 V =53248:REM A VIC KEZDOCIME
170 GT= 8192:REM A GRAFIKUS TAROLO KEZDOCIME
180 GOSUB 10000:GOSUB 10200
185 SZ=7*16+2:GOSUB 10400:REM ELOKESZITES
190 XR=PEEK(V+19):REM FENYC.X-RASZTERKOORD.-JA
200 YR=PEEK(V+20):REM FENYC.Y-RASZTERKOORD.-JA
210 X =2*(XR-40) :REM GRAFIKUS KOORDINATA
220 Y =YR-40      :REM GRAFIKUS KOORDINATA
230 IF X>319 OR X<0 OR Y>199 OR Y<0 THEN 190
235 REM ADATOK ELLENORZESE
240 GOSUB 10700:REM PONT BEKAPCSOLASA
250 GOTO 190
10000 REM **************************
10010 REM *                        *
10020 REM * A GRAFIKA BEKAPCSOLASA *
10030 REM *                        *
10040 REM **************************
10050 REM
10060 V=53248:REM A VIC KEZDOCIME
10070 POKE V+17,PEEK(V+17) OR (8+3)*16
10075 REM GRAFIKA BEKAPCSOLASA
10080 POKE V+22, PEEK(V+22) AND 255-16
10085 REM TOBBSZINU UM. KIKAPCSOLASA
10090 POKE V+24, PEEK(V+24) OR 8
10095 REM GRAFIKUS TAR A $2000(8192)-RE
10100 RETURN
10200 REM **********************
10210 REM *                    *
10220 REM * GR. TAROLO TORLESE *
10230 REM *                    *
10240 REM **********************
10250 REM
10260 GT=8192:REM A GRAFIKUS TAROLO KEZDOCIME
10270 FOR I=GT TO GT+8000:REM 8000 BYTE
10280 POKE I,0:REM TORLES
10290 NEXT I
10300 RETURN
```

```
10400 REM ********************
10410 REM *                  *
10420 REM * A SZINEK TORLESE *
10430 REM *                  *
10440 REM ********************
10450 REM
10460 KT=1024:REM A KEPERNYOTAROLO KEZDOCIME
10470 SZ=6*16+7:REM PONT=KEK HATTER=SARGA
10480 FOR I=KT TO KT+1000:REM 1000 BYTE
10490 POKE I,SZ:REM A SZINKODOK BEIRASA
10500 NEXT I
10510 RETURN
10700 REM *********************
10710 REM *                   *
10720 REM * PONT BEKAPCSOLASA *
10730 REM *                   *
10740 REM *********************
10750 REM
10760 RA=320*INT(Y/8) + (Y AND 7)
10770 BA=8*INT(X/8)
10780 MA=2↑(7-(X AND 7))
10790 AD=GT+RA+BA
10800 POKE AD,PEEK(AD) OR MA
10805 RETURN
```

A grafika bekapcsolása és a grafikus tár törlése után a 190-es és 200-as sorban kiolvassuk a fényceruzával megjelölt képernyői pont koordinátáit. Ezekből az általunk ismert képlettel kiszámítjuk a grafikus koordinátákat (210-es, 220-as sor). Ha megkaptuk az eredményt, ellenőriznünk kell, hogy a pont a képernyőablakon belül van-e (230-as sor). Csak ezután ugorhatunk a pont elhelyezését végző alprogramba (10700-as sor).

A program, fantáziánktól függően, tovább bővíthető. Például az egyenes végpontjainak, vagy a kör középpontjának és sugarának kijelölése után "gombnyomásra" megrajzolhatjuk az egyenest vagy a kört. A fényceruzával a képernyőn megjelenő menüpontok közül is választhatunk.

De térjünk vissza a programhoz. Teljesen mindegy ebben az esetben, hogy a fényceruza a képernyőre mutat vagy nem, egy pont mindenképpen a képernyőre kerül. Ez pillanatnyilag nem jelent problémát, de esetenként zavaró lehet. Említettük már, hogy a fényceruzával is váltható ki megszakítás, de csak akkor, ha az jelzést küld a számítógépnek, vagyis ha ténylegesen a képernyőre mutat.

Érdekes alkalmazási lehetőség a következő. A fényceruza és a számítógép segítségével riasztóberendezést készíthetünk. Szereljük fel a fényceruzánkat a szobai lámpára. Amikor valaki belép és felkapcsolja a villanyt, a fényceruza impulzust küld a számítógépnek, ami rögtön úgy elkezd szirénázni, hogy még

a szomszédok is kiugranak az ágyból. Továbbfejleszthetjük úgy, hogy a lámpát az ajtó nyitása működtesse. Ezt azért most nem valósítjuk meg, helyette csak a keret színváltására mutatunk példát:

```
                               ;*** P 36 ***
                               ;
100:     C800                           *=     $C800    ;KEZDOCIM
110:     C800              SZIN         =      $FB
120:     C800              IRQVECT      =      $0314
130:     C800              IRQREGI      =      $EA31
140:     C800              IRR          =      $D019
150:     C800              IMR          =      $D01A
160:     C800              KERET        =      $D020
                               ;
                               ;INICIALIZALAS
                               ;*************
                               ;
210:     C800 78           INIC         SEI             ;MEGSZAKITAS TILTASA
220:     C801 A9 16                     LDA    #<IRQUJ
230:     C803 8D 14 03                  STA    IRQVECT
240:     C806 A9 C8                     LDA    #>IRQUJ
250:     C808 8D 15 03                  STA    IRQVECT+1 ;IRQ-VEKTOR ATIRASA
260:     C80B A9 00                     LDA    #$00
270:     C80D 85 FB                     STA    SZIN     ;SZIN FEKETE
280:     C80F A9 88                     LDA    #%10001000 ;MASZK
290:     C811 8D 1A D0                  STA    IMR      ;FENYCERUZA IRQ KIVALASZTA
300:     C814 58                        CLI             ;MEGSZAKITAS ENGEDELYEZESE
310:     C815 60                        RTS
                               ;
                               ;MEGSZAKITASI RUTIN
                               ;******************
                               ;
360:     C816 AD 19 D0     IRQUJ        LDA    IRR      ;MEGSZAKITAS REGISZTER
370:     C819 8D 19 D0                  STA    IRR      ;TORLESE
380:     C81C 30 07                     BMI    IRQRAS   ;RASTERSOR IRQ
                               ;NORMAL IRQ
400:     C81E AD 0D DC                  LDA    $DC0D    ;CIA 1-IRR TORLESE
410:     C821 58                        CLI             ;MEGSZAKITAS ENGEDELYEZESE
420:     C822 4C 31 EA                  JMP    IRQREGI  ;A REGI RUTINRA
430:     C825 A5 FB        IRQRAS       LDA    SZIN
440:     C827 8D 20 D0                  STA    KERET    ;KERETSZIN
450:     C82A E6 FB                     INC    SZIN     ;SZINKOD NOVELESE
460:     C82C 4C BC FE                  JMP    $FEBC    ;BEFEJEZES
```

A program BASIC-betöltő listája a következő:

```
1 REM *** P 36 B ***
2 REM
100 FOR I=51200 TO 51246
110 READ X:POKE I,X:S=S+X:NEXT I
120 DATA 120,169, 22,141, 20,  3
```

```
125 DATA 169,200,141, 21,  3,169
130 DATA   0,133,251,169,136,141
135 DATA  26,208, 88, 96,173, 25
140 DATA 208,141, 25,208, 48,  7
145 DATA 173, 13,220, 86, 76, 49
150 DATA 234,165,251,141, 32,208
155 DATA 230,251, 76,188,254
160 IF S<>5910 THEN PRINT"HIBA A DATA-BAN !":END
170 PRINT"RENDBEN !"
```

A program SYS 51200-zal indítható.

A következőkben megadott sorszámok az assemblerlistára vonatkoznak.

A 220-as sorban kezdődő inicializáló rutin már ismerős. Először is átállítjuk saját rutinunkra a megszakítási mutatót, majd az IMR 3. bitjének bekapcsolásával kiválasztjuk a fényceruza-megszakítást. A megszakítási rutin a 360-as sortól kezdődik és nem jelent semmiféle újdonságot. Megállapítjuk a megszakítás jellegét (380-as sor), majd az eredménytől függően elágaztatjuk a programot a saját vagy az eredeti megszakítási rutinra. A saját megszakítási rutin végrehajtása a keretszín megváltozását eredményezi.

Maga a folyamat könnyen érthető. Lényege az, hogy egy tárrekesz tartalmát a megszakítási rutin által módosítjuk. Különleges hatásokat érhetünk el vele, de arra mindig vigyázzunk, hogy az itt lejátszódó folyamatokat pontosan követni tudjuk, mert megszakítást bármikor, bármelyik rutin kiválthat.

A megszakítási rutin pillanatnyi állapotát kívülről, pl. BASIC-ből is lekérdezhetjük, vagy egyéb vezérlési funkciókat is átvehetünk. Ezt az operációs rendszer is kihasználja. A TI$-órát például, amit a belső megszakítási rutin léptet, BASIC-ből bármikor lekérdezhetjük.

Ennyit a programról. Emlékeztetőül még csak annyit, hogy ez a képernyővillogtatás az 1-es vezérlőporton keresztül a tűzgombbal is előidézhető.

A megszakítás tehát egy olyan "játékszer", amit rendkívül sokoldalúan használhatunk, ha megfelelően tudunk bánni vele. Kísérletezzünk, érdemes!

4.7 Kis grafikai programcsomag

Eddig is már nagyon sokat hallottunk és olvastunk a grafikáról és programozásáról. A sok bemutatott rutin megkönnyíti a munkánkat és bepillantást enged a képek fantasztikus világába. A kíváncsian bevitt programok

eredményében nem csalódtunk, de annál inkább az odavezető útban. Az időközben kinőtt szakállon és megőszült hajon igazán nem lehet csodálkozni. BASIC-programjaink még akkor is lassúak, ha a Függelék útmutatásai alapján a lehető legoptimálisabb megoldásokat választottuk.

Röviden: mielőtt bárki elvesztené a kedvét és türelmét, átnyújtunk most egy gépi nyelven írt grafikai programcsomagot, ami gyorssá és érdekessé teheti az egyébként elég gyötrelmes grafikai munkát. Ne rémüljünk meg, ha a következő oldalakat átlapozzuk. Tudjuk, hogy a gépi nyelvű program bevitele és ellenőrzése nem egyszerű dolog, de az biztos, hogy megéri! Aki a grafikával komolyan akar foglalkozni, ne riadjon vissza szabadidejének feláldozásától sem. Választási lehetőségeink a következők:

1. Megmaradunk az egyszerű, de lassú BASIC-programoknál.
2. Rászánunk egy napot és begépeljük a következő programot.
3. Szerzünk valamilyen grafikai bővítőprogramot, amivel rengeteg munkát takaríthatunk meg.
4. Beszerezzük a könyvhöz tartozó programlemezt, amelyen a könyvben közölt összes program, futtatásra kész állapotban megtalálható.

Itt kell azonban azt is megjegyeznünk, hogy a teljesen kezdőknek a grafikai munkát nem javasoljuk.

Ezek után térjünk a tárgyra.

Először a programcsomag assemblerlistáját közöljük, amit az érdeklődők számára bőven elláttunk magyarázatokkal. A programrészek szervezésekor néhány esetben eltértünk a már korábban közölt BASIC-programoktól. Ezt a kedvezőbb futási sebesség indokolta. Ilyen például az egyenes rajzolását végző rutin. Ebben a programban nem a végpontok koordinátáit számoljuk ki, amiből vissza kellene számolnunk a tárcímeket, hanem a fel/le és jobbra/balra megtett lépések számát a kezdőponttól a végpontig. Közben arra is ügyeljünk, hogy mindig két-két pont legyen egymás mellett, ill. alatt, hogy folyamatos vonalat kapjunk.

Az assemblerlista után a BASIC-betöltő programot is közöljük, ami lehetővé teszi, hogy a programot monitorprogram nélkül is biztonságosan betöltsük.

```
                ;*** P 37 ***
                ;
                ;
                ;
C800                    *=      $C800   ;KEZDOCIM
                ;
                ; ********************************
                ; **                            **
                ; **  NAGYFELBONTASU GRAFIKUS   **
                ; **       PROGRAMCSOMAG        **
                ; **                            **
                ; ********************************
                ;
                ;
                ;
                ;ROM-UGROCIMEK
                ;*************
                ;
C800            GETBYT  =       $B79E   ;BYTE ERTEK BEOLVASAS
C800            CHKCOM  =       $AEFD   ;ELLENORZES EGESZRE
C800            CHKGET  =       $B7F1   ;CHKCOM+GETBYT
C800            GETCOR  =       $B7EB   ;KOORDINATAK BEOLVASASA
C800            QERR    =       $B248   ;ILLEGAL QUANTITY KIIRATASA
C800            V       =       $D000   ;VIDEOVEZERLO
                ;
                ;NULLASLAP REGISZTEREK
                ;*********************
                ;
C800            OFFX    =       $63
C800            MSK     =       $AB     ;MASZK
C800            DIF0    =       $69
C800            DIF1    =       $6A
C800            DIF2    =       $6B
C800            DIF3    =       $6C
C800            DIF4    =       $6D
C800            DIF5    =       $6E
C800            ZWN     =       $6F     ;XK/8(FELSO SOR)
C800            ZA      =       $70     ;SZAMLALO(FELSO SOR)
C800            A       =       $AC     ;PONTCIM
C800            B       =       $AC+1
C800            XK      =       $14     ;X-KOORDINATA
C800            FLG     =       $97     ;TORLES/RAJZOLAS-JELZO
C800            USE     =       $FD     ;DIVERSES
                ;
                ;ALLANDO ERTEKEK
                ;***************
                ;
C800            GRAF    =       $2000   ;GRAFIKUS TAR KEZDOCIME
C800            VIDEO   =       $0400   ;VIDEORAM KEZDOCIME
C800            GSTART  =       $21     ;GR-START+1-FELSO BYTE
C800            GEND    =       $3E     ;GR-ENDE-1-FELSO BYTE
                ;
```

```
                            ;
                            ;VEZERLO CIMEK
                            ;*************
                            ;
C800 4C 24 C8                       JMP  INIC     ;GRAFIKA BE
C803 4C 41 C8                       JMP  GOFF     ;GRAFIKA KI
C806 4C 12 C9                       JMP  GCLEAR   ;GRAFIKA TORLESE
C809 4C 31 C9                       JMP  SSZIN    ;SZIN BEALLITASA(TORLESE)
C80C 4C 2A C9                       JMP  PSZIN    ;RAJZOLAS SZINE
C80F 4C 58 C8                       JMP  RAJZ     ;PONT RAJZOLASA
C812 4C 55 C8                       JMP  TORLES   ;PONT TORLESE
C815 4C 6B C8                       JMP  SLINE    ;EGYENES RAJZOLASA
C818 4C 68 C8                       JMP  CLLINE   ;EGYENES TORLESE
C81B 4C 43 CA                       JMP  GLOAD    ;GRAFIKA BETOLTESE
C81E 4C 52 CA                       JMP  GSAVE    ;GRAFIKA TAROLASA
C821 4C 69 CA                       JMP  HARDC    ;HARDCOPY (GP 100 VC STB)
                            ;
                            ;
                            ;GRAFIKA BEKAPCSOLASA
                            ;********************
                            ;
C824 EA            INIC             NOP           ;NINCS MUVELETVEGZES
C825 AD 11 D0                       LDA  V+17
C828 8D 1E CB                       STA  STORE1
C82B AD 18 D0                       LDA  V+24
C82E 8D 1F CB                       STA  STORE2   ;REGI TARTALOM MENTESE
C831 A9 3B                          LDA  #%00111011
C833 8D 11 D0                       STA  V+17     ;GRAFIKA BE
C836 A9 18                          LDA  #%00011000
C838 8D 18 D0                       STA  V+24     ;A $2000 CIMRE
C83B A9 60                          LDA  #$60     ;A TOBBSZOROS INICIALIZAL
C83D 8D 24 C8                       STA  INIC     ;MEGAKADALYOZASA
C840 60                             RTS
                            ;
                            ;
                            ;GRAFIKA KIKAPCSOLASA
                            ;********************
                            ;
C841 AD 1E CB      GOFF             LDA  STORE1
C844 8D 11 D0                       STA  V+17
C847 AD 1F CB                       LDA  STORE2
C84A 8D 18 D0                       STA  V+24     ;REGI TARTALOM VISSZAIRAS
C84D A9 EA                          LDA  #$EA     ;CODE FUER NOP FUER
C84F 8D 24 C8                       STA  INIC     ;BLOCKADE AUFHEBEN
C852 4C 44 E5                       JMP  $E544    ;KEPERNYO TORLESE
                            ;
                            ;
                            ;PONT TORLESE
                            ;************
                            ;
C855 A2 00         TORLES           LDX  #$00     ;TORLES JELZO
C857 2C                             .BYTE$2C
                            ;
```

```
                          ;
                          ;PONT RAJZOLASA
                          ;*************
                          ;
1080:   C858 A2 80     RAJZ     LDX   #$80      ;RAJZOLAS JELZO
1090:   C85A 86 97     PL1      STX   FLG
1100:   C85C 20 FD AE           JSR   CHKCOM
1110:   C85F 20 79 C8           JSR   TESCOR    ;KOORDINATAK BEOLVASASA
1120:   C862 20 94 C8           JSR   POZSZ     ;CIM KISZAMITASA
1130:   C865 4C E2 C8           JMP   PLT       ;PONT RAJZOLASA/TORLESE
                          ;
                          ;
                          ;EGYENES TORLESE
                          ;**************
                          ;
1190:   C868 A2 00     CLLINE   LDX   #$00      ;TORLES JELZO
1200:   C86A 2C                 .BYTE$2C
                          ;
                          ;
                          ;EGYENES RAJZOLASA
                          ;****************
                          ;
1260:   C86B A2 80     SLINE    LDX   #$80      ;RAJZOLAS JELZO
1270:   C86D 20 5A C8           JSR   PL1       ;ELSO PONT RAJZOLASA
1280:   C870 20 FD AE           JSR   CHKCOM
1290:   C873 20 79 C8           JSR   TESCOR    ;MASODIK KOORDINATA BE
1300:   C876 4C BB C9           JMP   HLINE     ;EGYENES RAJZOLASA
                          ;
                          ;
                          ;KORDINATAK ELLENORZESE
                          ;*********************
                          ;
1360:   C879 20 EB B7  TESCOR   JSR   GETCOR    ;BEOLVASAS
1370:   C87C 8A                 TXA
1380:   C87D A8                 TAY
1390:   C87E A2 15              LDX   #XK+1
1400:   C880 C0 C8              CPY   #200      ;Y-KOORD>=200?
1410:   C882 B0 0D              BCS   ILLFF     ;IGEN!
1420:   C884 A5 14              LDA   XK
1430:   C886 E0 01              CPX   #>320     ;X-KOORD>=320?
1440:   C888 90 06              BCC   T1        ;IGEN
1450:   C88A D0 05              BNE   ILLFF     ;A,XK-ALSO
1460:   C88C C9 40              CMP   #<320     ;X,XK-FELSO
1470:   C88E B0 01              BCS   ILLFF     ;Y,YK
1480:   C890 60        T1       RTS
1490:   C891 4C 48 B2  ILLFF    JMP   QERR      ;ILLEGAL QUANTITY
                          ;
                          ;
                          ;CIM KISZAMITASA
                          ;**************
                          ;
1550:   C894 8C 1C CB  POZSZ    STY   YK        ;Y-K
1560:   C897 8D 1A CB           STA   XKL       ;X-KL
1570:   C89A 8E 1B CB           STX   XKH       ;X-KH (KOZBENSO TAROLO)
1580:   C89D 85 14              STA   XK
1590:   C89F 86 15              STX   XK+1
```

```
1600:   C8A1 98                 TYA
1610:   C8A2 4A                 LSR  A
1620:   C8A3 4A                 LSR  A
1630:   C8A4 4A                 LSR  A        ;INT(Y/8)
1640:   C8A5 AA                 TAX
1650:   C8A6 BD 2F CB           LDA  MULH,X   ;320*INT(Y/8) (FELSO-BYTE)
1660:   C8A9 85 AD              STA  B
1670:   C8AB 8A                 TXA
1680:   C8AC 29 03              AND  #3       ;0-1 BIT MASZKOLASA
1690:   C8AE AA                 TAX
1700:   C8AF BD 49 CB           LDA  MULL,X   ;320*INT(Y/8) (ALSO-BYTE)
1710:   C8B2 85 AC              STA  A
1720:   C8B4 98                 TYA           ;Y-KOORD
1730:   C8B5 29 07              AND  #7       ;(Y AND 7)
1740:   C8B7 18                 CLC
1750:   C8B8 65 AC              ADC  A        ;OFFY=320*INT(Y/8)+(Y AND 7)
1760:   C8BA 85 AC              STA  A        ;****************************
1770:   C8BC A5 14              LDA  XK
1780:   C8BE 29 FB              AND  #$FB     ;OFFX=8*INT(X/8)
1790:   C8C0 85 63              STA  OFFX     ;***************
1800:   C8C2 A9 20              LDA  #>GRAF   ;GRAFIKUS OLDAL
1810:   C8C4 05 AD              ORA  B
1820:   C8C6 85 AD              STA  B        ;+SA
1830:   C8C8 18                 CLC
1840:   C8C9 A5 AC              LDA  A
1850:   C8CB 65 63              ADC  OFFX     ;AD=OFFY+OFFX+SA
1860:   C8CD 85 AC              STA  A        ;******************
1870:   C8CF A5 AD              LDA  B
1880:   C8D1 65 15              ADC  XK+1
1890:   C8D3 85 AD              STA  B
1900:   C8D5 A5 14              LDA  XK
1910:   C8D7 29 07              AND  #7
1920:   C8D9 49 07              EOR  #7       ;7-(X AND 7)
1930:   C8DB AA                 TAX
1940:   C8DC BD 4D CB           LDA  MSKTAB,X ;MASZK TABLAZAT
1950:   C8DF 85 AB              STA  MSK      ;2↑(7-(X AND 7))
1960:   C8E1 60                 RTS
                        ;
                        ;
                        ;PONT RAJZOLASA
                        ;**************
                        ;
2020:   C8E2 A0 00      PLT     LDY  #0
2030:   C8E4 08                 PHP           ;CARRY-JELZO(FLAG) MENTESE
2040:   C8E5 A5 AB              LDA  MSK      ;MASZK
2050:   C8E7 24 97              BIT  FLG      ;RAJZOLAS/TORLES JELZO
2060:   C8E9 30 05              BMI  PL2      ;RAJZOLAS
2070:   C8EB 49 FF              EOR  #$FF     ;TORLES
2080:   C8ED 31 AC              AND  (A),Y
2090:   C8EF 2C                 .BYTE$2C      ;UGRAS A KOVETKEZO UTASITAS
2100:   C8F0 11 AC      PL2     ORA  (A),Y    ;RAJZOLAS
2110:   C8F2 91 AC              STA  (A),Y
2120:   C8F4 A5 AC              LDA  A        ;SZIN BEALLITASA
2130:   C8F6 85 FD              STA  USE
2140:   C8F8 A5 AD              LDA  B
```

```
C8FA 4A                   LSR  A
C8FB 66 FD                ROR  USE
C8FD 4A                   LSR  A
C8FE 66 FD                ROR  USE
C900 4A                   LSR  A
C901 66 FD                ROR  USE       ;CIM/8
C903 29 03                AND  #$03
C905 09 04                ORA  #$04      ;VIDEORAM $0400-TOL
C907 85 FE                STA  USE+1
C909 AD 1D CB             LDA  SZIN      ;SZIN
C90C 91 FD                STA  (USE),Y   ;A VIDEORAMBA
C90E 28                   PLP            ;CARRY-JELZO(FLAG)
C90F A4 6F                LDY  ZWN
C911 60                   RTS
                 ;
                 ;
                 ;GRAFIKA TORLESE
                 ;****************
                 ;
C912 A9 20       GCLEAR   LDA  #>GRAF
C914 85 FE                STA  USE+1
C916 A0 00                LDY  #<GRAF    ;Y=0
C918 84 FD                STY  USE       ;GRAFIKA KEZDOCIME
C91A A2 20                LDX  #$20      ;HOSSZ
C91C 98                   TYA            ;Y=0!
C91D 91 FD       GC1      STA  (USE),Y   ;TORLES
C91F C8                   INY
C920 D0 FB                BNE  GC1
C922 E6 FE                INC  USE+1
C924 CA                   DEX
C925 D0 F6                BNE  GC1
C927 4C 37 C9             JMP  SCOL1     ;VIDEORAM TORLESE
                 ;
                 ;
                 ;PONTSZIN DEFINIALASA
                 ;********************
                 ;
C92A 20 F1 B7 PSZIN       JSR  CHKGET    ;EGESZ+SZIN
C92D 8E 1D CB             STX  SZIN
C930 60                   RTS
                 ;
                 ;
                 ;SZIN BEALLITASA
                 ;***************
                 ;
C931 20 F1 B7 SSZIN       JSR  CHKGET    ;EGESZ+SZIN
C934 8E 1D CB             STX  SZIN
C937 A2 03       SCOL1    LDX  #3
C939 A9 04                LDA  #>VIDEO
C93B 85 FE                STA  USE+1
C93D A0 00                LDY  #<VIDEO
C93F 84 FD                STY  USE       ;VIDEORAM CIME
C941 84 97                STY  FLG       ;Y=0!
C943 AD 1D CB             LDA  SZIN      ;SZIN
C946 91 FD       GM9      STA  (USE),Y   ;BEALLITAS
```

```
2700:   C948 C8                  INY
2710:   C949 C4 97               CPY   FLG
2720:   C94B D0 F9               BNE   GM9
2730:   C94D E6 FE               INC   USE+1
2740:   C94F CA                  DEX
2750:   C950 F0 03               BEQ   GM90
2760:   C952 10 F2               BPL   GM9
2770:   C954 60                  RTS
2780:   C955 A2 E8      GM90     LDX   #$E8
2790:   C957 86 97               STX   FLG         ;SPRITE-MUTATO TAROLASA
2800:   C959 D0 EB               BNE   GM9
                        ;
                        ;
                        ;MUTATO RUTINOK
                        ;**************
                        ;
2860:   C95B A5 AC      ALSO     LDA   A           ;PONTCIM ALSO
2870:   C95D 29 07               AND   #7
2880:   C95F C9 07               CMP   #7
2890:   C961 F0 05               BEQ   UN1         ;HATAR!
2900:   C963 38                  SEC
2910:   C964 A9 00               LDA   #0
2920:   C966 B0 04               BCS   UN2         ;TULCSORDULAS(C=1!)
2930:   C968 A9 38      UN1      LDA   #$38
2940:   C96A E6 AD               INC   B           ;C=1!
2950:   C96C 65 AC      UN2      ADC   A
2960:   C96E 85 AC               STA   A
2970:   C970 A9 00               LDA   #0
2980:   C972 65 AD               ADC   B
2990:   C974 85 AD               STA   B
3000:   C976 60                  RTS
                        ;
3020:   C977 30 E2      AF       BMI   ALSO
                        ;
3040:   C979 A5 AC      FELSO    LDA   A           ;CIM
3050:   C97B 29 07               AND   #7
3060:   C97D F0 05               BEQ   OB1         ;HATAR(FELSO)
3070:   C97F 18                  CLC
3080:   C980 A9 FF               LDA   #$FF
3090:   C982 90 04               BCC   OB2         ;KIVONAS 1
3100:   C984 A9 C7      OB1      LDA   #$C7        ;KIVONAS 320+7=327
3110:   C986 C6 AD               DEC   B           ;C=1!
3120:   C988 65 AC      OB2      ADC   A
3130:   C98A 85 AC               STA   A
3140:   C98C A5 AD               LDA   B
3150:   C98E E9 00               SBC   #0
3160:   C990 85 AD               STA   B
3170:   C992 60                  RTS
3200:   C993 46 AB      JOBB     LSR   MSK         ;MASZK ELTOLASA
3210:   C995 90 0E               BCC   RE2
3220:   C997 66 AB               ROR   MSK
3230:   C999 A5 AC               LDA   A
3240:   C99B C8                  INY
3250:   C99C 18                  CLC
3260:   C99D 69 08               ADC   #8
```

```
C99F 85 AC            STA   A
C9A1 90 02            BCC   RE2
C9A3 E6 AD            INC   B        ;+8
C9A5 60         RE2   RTS
                ;
C9A6 10 EB      JB    BPL   JOBB
                ;
C9A8 06 AB      BAL   ASL   MSK
C9AA 90 0E            BCC   LI1
C9AC 26 AB            ROL   MSK
C9AE A5 AC            LDA   A
C9B0 88               DEY
C9B1 38               SEC
C9B2 E9 08      LI3   SBC   #8
C9B4 85 AC            STA   A
C9B6 B0 02            BCS   LI1
C9B8 C6 AD            DEC   B        ;-8
C9BA 60         LI1   RTS
                ;
                ;
                ;EGYENES RAJZOLASA
                ;*****************
                ;
C9BB 48         HLINE PHA            ;A,X2-ALSO BYTE
C9BC AD 1B CB         LDA   XKH      ;X,X2-FELSO BYTE
C9BF 4A               LSR   A        ;Y,Y2
C9C0 AD 1A CB         LDA   XKL      ;XKL,X1-ALSO BYTE
C9C3 6A               ROR   A        ;XKH,X1-ALSO BYTE
C9C4 4A               LSR   A        ;YK,Y1
C9C5 4A               LSR   A
C9C6 85 6F            STA   ZWN      ;X1/8 KOZBENSO TAROLO
C9C8 68               PLA
C9C9 48               PHA
C9CA 38               SEC
C9CB ED 1A CB         SBC   XKL
C9CE 48               PHA
C9CF 8A               TXA
C9D0 ED 1B CB         SBC   XKH
C9D3 85 6C            STA   DIF3     ;X2-X1
C9D5 B0 0A            BCS   L3       ;NEGATIV"?"
C9D7 68               PLA            ;IGEN
C9D8 49 FF            EOR   #$FF
C9DA 69 01            ADC   #1
C9DC 48               PHA
C9DD A9 00            LDA   #0
C9DF E5 6C            SBC   DIF3     ;ELOJELVALTAS
C9E1 85 6A      L3    STA   DIF1
C9E3 85 6E            STA   DIF5     ;(X2-X1)-FELSO
C9E5 68               PLA
C9E6 85 69            STA   DIF0
C9E8 85 6D            STA   DIF4     ;(X2-X1)-ALSO
C9EA 68               PLA
C9EB 8D 1A CB         STA   XKL
C9EE 8E 1B CB         STX   XKH
C9F1 98               TYA
```

```
C9F2 18                   CLC
C9F3 ED 1C CB             SBC  YK       ;Y2-Y1
C9F6 90 04                BCC  L4       ;NEGATIV"?"
C9F8 49 FF                EOR  #$FF
C9FA 69 FE                ADC  #$FE     ;ELOJELVALTAS.
C9FC 85 6B      L4        STA  DIF2     ;(Y2-Y1)
C9FE 8C 1C CB             STY  YK
CA01 66 6C                ROR  DIF3     ;(X2-X1)/2
CA03 38                   SEC
CA04 E5 69                SBC  DIF0     ;(Y2-Y1)-(X2-X1)
CA06 AA                   TAX           ;ALSO BYTE AZ X REGISZTRE
CA07 A9 FF                LDA  #$FF
CA09 E5 6A                SBC  DIF1
CA0B 85 70                STA  ZA       ;FELSO BYTE ZA-BA
CA0D A4 6F                LDY  ZWN
CA0F B0 05                BCS  L5
CA11 0A         L1        ASL  A
CA12 20 A6 C9             JSR  JB       ;JOBB/BAL
CA15 38                   SEC
CA16 A5 6D      L5        LDA  DIF4
CA18 65 6B                ADC  DIF2
CA1A 85 6D                STA  DIF4
CA1C A5 6E                LDA  DIF5
CA1E E9 00                SBC  #0       ;(X2-X1)-(Y2-Y1)NACH(X2-
CA20 85 6E      L2        STA  DIF5
CA22 84 6F                STY  ZWN
CA24 20 E2 C8             JSR  PLT      ;PONT RAJZOLASA
CA27 E8                   INX
CA28 D0 05                BNE  L6
CA2A E6 70                INC  ZA
CA2C D0 01                BNE  L6       ;DEC SZAMOK
CA2E 60                   RTS
CA2F A5 6C      L6        LDA  DIF3
CA31 B0 DE                BCS  L1
CA33 20 77 C9             JSR  AF       ;LE/FEL
CA36 18                   CLC
CA37 A5 6D                LDA  DIF4
CA39 65 69                ADC  DIF0
CA3B 85 6D                STA  DIF4
CA3D A5 6E                LDA  DIF5
CA3F 65 6A                ADC  DIF1
CA41 50 DD                BVC  L2
                ;
                ;
                ;GRAFIKA BETOLTESE
                ;*****************
                ;
CA43 20 FD AE GLOAD       JSR  CHKCOM   ;EGESZ
CA46 20 D4 E1             JSR  $E1D4    ;PARAMETEREK BEOLVASASA
CA49 A0 20                LDY  #>GRAF
CA4B A2 00                LDX  #<GRAF   ;KEZDOCIM
CA4D A9 00                LDA  #$00     ;LOAD-JELZO
CA4F 4C D5 FF             JMP  $FFD5    ;BETOLTES
                ;
```

```
                               ;
                               ;GRAFIKA TAROLASA
                               ;****************
                               ;
4400:  CA52 20 FD AE GSAVE     JSR  CHKCOM   ;EGESZ"?"
4410:  CA55 20 D4 E1           JSR  $E1D4
                               ;VEGCIM
4430:  CA58 A2 3F              LDX  #>GRAF+8000
4440:  CA5A A0 40              LDY  #<GRAF+8000
4450:  CA5C A9 00              LDA  #<GRAF
4460:  CA5E 85 FD              STA  $FD
4470:  CA60 A9 20              LDA  #>GRAF
4480:  CA62 85 FE              STA  $FE      ;KEZDOCIM
4490:  CA64 A9 FD              LDA  #$FD     ;MUTATO
4500:  CA66 4C D8 FF           JMP  $FFD8    ;TAROLAS
                               ;
                               ;
                               ;HARDCOPY SEIKOSHA GP 100 VC
                               ;***************************
                               ;
4560:  CA69 20 F1 B7 HARDC     JSR  CHKGET   ;LOG FILESZAM
4570:  CA6C 86 67              STX  $67
4580:  CA6E 20 0F F3           JSR  $F30F    ;LOG FILESZAM KERESESE
4590:  CA71 20 1F F3           JSR  $F31F    ;FILEPARAMET-EK BEALLITASA
4600:  CA74 A6 67              LDX  $67
4610:  CA76 20 C9 FF           JSR  $FFC9    ;CSATORNA MEGNYITASA
4620:  CA79 A9 FF              LDA  #$FF
4630:  CA7B 85 61              STA  $61      ;MASZK
4640:  CA7D A9 07              LDA  #7
4650:  CA7F 85 FD              STA  USE      ;NAGYSAG
4660:  CA81 A9 1C              LDA  #28
4670:  CA83 85 97              STA  FLG      ;SORSZAMLALO
4680:  CA85 A9 00              LDA  #0
4690:  CA87 8D 20 CB           STA  KOZB
4700:  CA8A A9 28     HA1      LDA  #40
4710:  CA8C 8D 21 CB           STA  FLG2     ;SZAMLALO
4720:  CA8F A2 04              LDX  #4
4730:  CA91 BD 2A CB HA10      LDA  HATAB,X  ;KOZPONTOZAS
4740:  CA94 20 D2 FF           JSR  $FFD2    ;KIVITEL
4750:  CA97 CA                 DEX
4760:  CA98 10 F0              BPL  HA1
4770:  CA9A A9 00              LDA  #0
4780:  CA9C 85 63              STA  $63
4790:  CA9E 85 64              STA  $64      ;XK=0
4800:  CAA0 AD 20 CB HA2       LDA  KOZB
4810:  CAA3 85 65              STA  $65      ;YK
4820:  CAA5 A9 00              LDA  #0
4830:  CAA7 85 FE              STA  USE+1    ;PUFFERMUTATO
4840:  CAA9 A5 63     HA3      LDA  $63      ;XK-L
4850:  CAAB A6 64              LDX  $64      ;XK-H
4860:  CAAD A4 65              LDY  $65      ;YK
4870:  CAAF 20 94 C8           JSR  POZSZ    ;POSICIO SZAMITAS
4880:  CAB2 A0 00              LDY  #0
4890:  CAB4 B1 AC              LDA  (A),Y    ;BYTE BEOLVASAS
4900:  CAB6 A6 FE              LDX  USE+1    ;PUFFERMUTATO
4910:  CAB8 9D 22 CB           STA  PUFF,X   ;A PUFFERBA
```

```
4920:   CABB E6 65           INC  $65      ;YK
4930:   CABD E8              INX
4940:   CABE 86 FE           STX  USE+1
4950:   CAC0 E4 FD           CPX  USE
4960:   CAC2 D0 E5           BNE  HA3
4970:   CAC4 A9 00           LDA  #0
4980:   CAC6 A0 07           LDY  #7
4990:   CAC8 A6 FD     HA4   LDX  USE
5000:   CACA 1E 22 CB HA5    ASL  PUFF,X   ;EGY BIT BEOLVASASA
5010:   CACD 2A              ROL  A        ;ES AZ AKKUBA TOLTESE
5020:   CACE CA              DEX           ;PUFFERMUTATO NOVELESE
5030:   CACF 10 F9           BPL  HA5
5040:   CAD1 25 61           AND  $61      ;UTOLSO SORNAL=#$0F
5050:   CAD3 09 80           ORA  #$80     ;FELSO BYTE
5060:   CAD5 20 D2 FF        JSR  $FFD2    ;UGRAS
5070:   CAD8 88              DEY
5080:   CAD9 10 ED           BPL  HA4
5090:   CADB A5 63           LDA  $63      ;XK-L
5100:   CADD 18              CLC
5110:   CADE 69 08           ADC  #8
5120:   CAE0 85 63           STA  $63      ;XK+8
5130:   CAE2 90 02           BCC  HA6
5140:   CAE4 E6 64           INC  $64      ;XK-H
5150:   CAE6 CE 21 CB HA6    DEC  FLG2
5160:   CAE9 D0 B5           BNE  HA2
5170:   CAEB A9 0D           LDA  #$0D     ;CARRIAGE RETURN
5180:   CAED 20 D2 FF        JSR  $FFD2    ;UGRAS
5190:   CAF0 AD 20 CB        LDA  KOZB
5200:   CAF3 18              CLC
5210:   CAF4 69 07           ADC  #7
5220:   CAF6 8D 20 CB        STA  KOZB     ;YK+7
5230:   CAF9 C6 97           DEC  FLG
```

És a BASIC-betöltő lista:

```
1 REM *** P 37 B ***
2 REM
100 FOR I=51200 TO 52054
110 READ X:POKE I,X:S=S+X:NEXT I
120 DATA  76, 36,200, 76, 65,200
125 DATA  76, 18,201, 76, 49,201
130 DATA  76, 42,201, 76, 88,200
135 DATA  76, 85,200, 76,107,200
140 DATA  76,104,200, 76, 67,202
145 DATA  76, 82,202, 76,105,202
150 DATA 234,173, 17,208,141, 30
155 DATA 203,173, 24,208,141, 31
160 DATA 203,169, 59,141, 17,208
165 DATA 169, 24,141, 24,208,169
170 DATA  96,141, 36,200, 96,173
175 DATA  30,203,141, 17,208,173
180 DATA  31,203,141, 24,208,169
```

```
185 DATA 234,141, 36,200, 76, 68
190 DATA 229,162,  0, 44,162,128
195 DATA 134,151, 32,253,174, 32
200 DATA 121,200, 32,148,200, 76
205 DATA 226,200,162,  0, 44,162
210 DATA 128, 32, 90,200, 32,253
215 DATA 174, 32,121,200, 76,187
220 DATA 201, 32,235,183,138,168
225 DATA 166, 21,192,200,176, 13
230 DATA 165, 20,224,  1,144,  6
235 DATA 208,  5,201, 64,176,  1
240 DATA  96, 76, 72,178,140, 28
245 DATA 203,141, 26,203,142, 27
250 DATA 203,133, 20,134, 21,152
255 DATA  74, 74, 74,170,189, 47
260 DATA 203,133,173,138, 41,  3
265 DATA 170,189, 73,203,133,172
270 DATA 152, 41,  7, 24,101,172
275 DATA 133,172,165, 20, 41,248
280 DATA 133, 99,169, 32,  5,173
285 DATA 133,173, 24,165,172,101
290 DATA  99,133,172,165,173,101
295 DATA  21,133,173,165, 20, 41
300 DATA   7, 73,  7,170,189, 77
305 DATA 203,133,171, 96,160,  0
310 DATA   8,165,171, 36,151, 48
315 DATA   5, 73,255, 49,172, 44
320 DATA  17,172,145,172,165,172
325 DATA 133,253,165,173, 74,102
330 DATA 253, 74,102,253, 74,102
335 DATA 253, 41, 3,  9,  4,133
340 DATA 254,173, 29,203,145,253
345 DATA  40,164,111, 96,169, 32
350 DATA 133,254,160,  0,132,253
355 DATA 162, 32,152,145,253,200
360 DATA 208,251,230,254,202,208
365 DATA 246, 76, 55,201, 32,241
370 DATA 183,142, 29,203, 96, 32
375 DATA 241,183,142, 29,203,162
380 DATA   3,169,  4,133,254,160
385 DATA   0,132,253,132,151,173
390 DATA  29,203,145,253,200,196
395 DATA 151,208,249,230,254,202
400 DATA 240,  3, 16,242, 96,162
405 DATA 232,134,151,208,235,165
410 DATA 172, 41,  7,201,  7,240
415 DATA   5, 56,169,  0,176,  4
420 DATA 169, 56,230,173,101,172
425 DATA 133,172,169,  0,101,173
430 DATA 133,173, 96, 48,226,165
435 DATA 172, 41,  7,240,  5, 24
440 DATA 169,255,144,  4,169,199
445 DATA 198,173,101,172,133,172
450 DATA 165,173,233,  0,133,173
```

```
455 DATA  96, 70,171,144, 14,102
460 DATA 171,165,172,200, 24,105
465 DATA   8,133,172,144,  2,230
470 DATA 173, 96, 16,235,  6,171
475 DATA 144, 14, 38,171,165,172
480 DATA 136, 56,233,  8,133,172
485 DATA 176,  2,198,173, 96, 72
490 DATA 173, 27,203, 74,173, 26
495 DATA 203,106, 74, 74,133,111
500 DATA 104, 72, 56,237, 26,203
505 DATA  72,138,237, 27,203,133
510 DATA 108,176, 10,104, 73,255
515 DATA 105,  1, 72,169,  0,229
520 DATA 108,133,106,133,110,104
525 DATA 133,105,133,109,104,141
530 DATA  26,203,142, 27,203,152
535 DATA  24,237, 28,203,144,  4
540 DATA  73,255,105,254,133,107
545 DATA 140, 28,203,102,108, 56
550 DATA 229,105,170,169,255,229
555 DATA 106,133,112,164,111,176
560 DATA   5, 10, 32,166,201, 56
565 DATA 165,109,101,107,133,109
570 DATA 165,110,233,  0,133,110
575 DATA 132,111, 32,226,200,232
580 DATA 208,  5,230,112,208,  1
585 DATA  96,165,108,176,222, 32
590 DATA 119,201, 24,165,109,101
595 DATA 105,133,109,165,110,101
600 DATA 106, 80,221, 32,253,174
605 DATA  32,212,225,160, 32,162
610 DATA   0,169,  0, 76,213,255
615 DATA  32,253,174, 32,212,225
620 DATA 162, 63,160, 64,169,  0
625 DATA 133,253,169, 32,133,254
630 DATA 169,253, 76,216,255, 32
635 DATA 241,183,134,103, 32, 15
640 DATA 243, 32, 31,243,166,103
645 DATA  32,201,255,169,255,133
650 DATA  97,169,  7,133,253,169
655 DATA  28,133,151,169,  0,141
660 DATA  32,203,169, 40,141, 33
665 DATA 203,162,  4,189, 42,203
670 DATA  32,210,255,202, 16,247
675 DATA 169,  0,133, 99,133,100
680 DATA 173, 32,203,133,101,169
685 DATA   0,133,254,165, 99,166
690 DATA 100,164,101, 32,148,200
695 DATA 160,  0,177,172,166,254
700 DATA 157, 34,203,230,101,232
705 DATA 134,254,228,253,208,229
710 DATA 169,  0,160,  7,166,253
715 DATA  30, 34,203, 42,202, 16
720 DATA 249, 37, 97,  9,128, 32
```

```
725 DATA 210,255,136, 16,237,165
730 DATA  99, 24,105,  8,133, 99
735 DATA 144,  2,230,100,206, 33
740 DATA 203,208,181,169, 13, 32
745 DATA 210,255,173, 32,203, 24
750 DATA 105,  7,141, 32,203,198
755 DATA 151,240,  3, 76,138,202
760 DATA 169,  4,197,253,240, 12
765 DATA 133,253,169,  1,133,151
770 DATA 169, 15,133, 97,208,235
775 DATA 169, 15, 32,210,255, 76
780 DATA 204,255, 48, 48, 48, 32
785 DATA  58, 32,130, 32, 88, 32
790 DATA  58, 32,143, 32, 87, 65
795 DATA  80,  0, 16, 27,  8,  0
800 DATA   1,  2,  3,  5,  6,  7
805 DATA   8, 10, 11, 12, 13, 15
810 DATA  16, 17, 18, 20, 21, 22
815 DATA  23, 25, 26, 27, 28, 30
820 DATA  31,  0, 64,128,192,  1
825 DATA   2,  4,  8, 16, 32, 64
830 DATA 128, 76, 79
840 IF S<>104883 THEN PRINT"HIBA A DATA-BAN !":END
850 PRINT"RENDBEN !"
```

Mind a 12 alprogram lehívása SYS-utasítással történik, esetenként néhány paraméterrel kiegészítve. Első pillantásra ez egy kicsit szokatlan, hiszen eddig úgy tudtuk, hogy a SYS a hivatkozási címen kívül más paramétert nem enged meg. Vannak azonban kivételes esetek és az új írásmódot mi is hamar meg fogunk szokni.

Legcélszerűbb, ha a DEMO-program (P38) szerint járunk el és a 12 változót, ill. hivatkozási címet a program elején rögzítjük, így a programban az alprogramok lehívása egyszerűen a változónevekkel történhet. A következő táblázatban felsoroltuk a különböző SYS-utasítások írásmódját és a hivatkozási címeket:

SYS 51200	– a grafika bekapcsolása,
SYS 51203	– a grafika kikapcsolása,
SYS 51206	– a grafikustár törlése,
SYS 51209, PSZ*16+HSZ	– színek beállítása,
SYS 51212, PSZ*16+HSZ	– színek módosítása,
SYS 51215, X, Y	– egy pont elhelyezése,
SYS 51218, X, Y	– egy pont törlése,
SYS 51221, X1, Y1, X2, Y2	– egyenes rajzolása,
SYS 51224, X1, Y1, X2, Y2	– egyenes törlése,

SYS 51227, "név", GA	– a grafika betöltése,
SYS 51230, "név", GA	– a grafika tárolása,
SYS 51233, LF	– hardcopy (S.GP–100 VC típ. nyomtatóra).

A változók jelentése:

PSZ	– a grafikus pont színkódja (0–15),
HSZ	– a háttérszín kódja (0–15),
X	– egy pont X koordinátája (0–319),
Y	– egy pont Y koordinátája (0–199),
X1/2	– egy egyenes kezdő- és végpontjának X koordinátája (0–319),
Y1/2	– egy egyenes kezdő- és végpontjának Y koordinátája (0–199),
GA	– készülékcím (kazetta=1, mágneslemez-meghajtó=8),
LF	– logikai file-szám (OPEN LF, GA).

Mielőtt rátérünk a program ismertetésére, meg kell beszélnünk azokat a legfontosabb dolgokat, amelyekkel a részletek a legegyszerűbben kipróbálhatók és tesztelhetők.

Az első három alprogram nem igényel magyarázatot. Azt kell tudni, hogy a grafika bekapcsolása nem, kikapcsolása viszont törli a képernyőt. A színek, ha közben nem módosítjuk őket, minden grafikus képnél azonosak lesznek, az eredeti beállításnak megfelelően. Ha a háttér színét zöldnek (színkódja 5), a pont színét pedig ibolyának (színkódja 4) választjuk, akkor HSZ=5, PSZ=4 lesz. A pontok színét a video-RAM byte-jai tárolják. A negyedik alprogram tehát a grafikai munka egészére vonatkozó, általános érvényű színbeállítást végzi. Ezt, egy-egy képre vonatkozóan az 5. alprogrammal módosíthatjuk. A 6. és 7. alprogram a képernyői pontok bekapcsolását és törlését végzi. Amikor ezekre hivatkozunk, a szín-RAM-nak már tartalmaznia kell a színekre vonatkozó információkat. Hasonlóan működik a 8. és 9. alprogram, amelyekkel a végpontjaival meghatározott egyenest a képernyőre rajzolhatjuk, ill. törölhetjük azt. A színeket ebben az esetben is előre be kell állítani.

A grafika tárolására és betöltésére vonatkozó alprogramokat ugyanúgy alkalmazzuk, mint egy egyszerű BASIC-program esetében. A SYS-utasítást ki kell egészítenünk a file-névvel és a készülékcímmel. Hardcopy készítéséhez először meg kell nyitnunk a nyomtatócsatornát:

```
OPEN 1,4
```

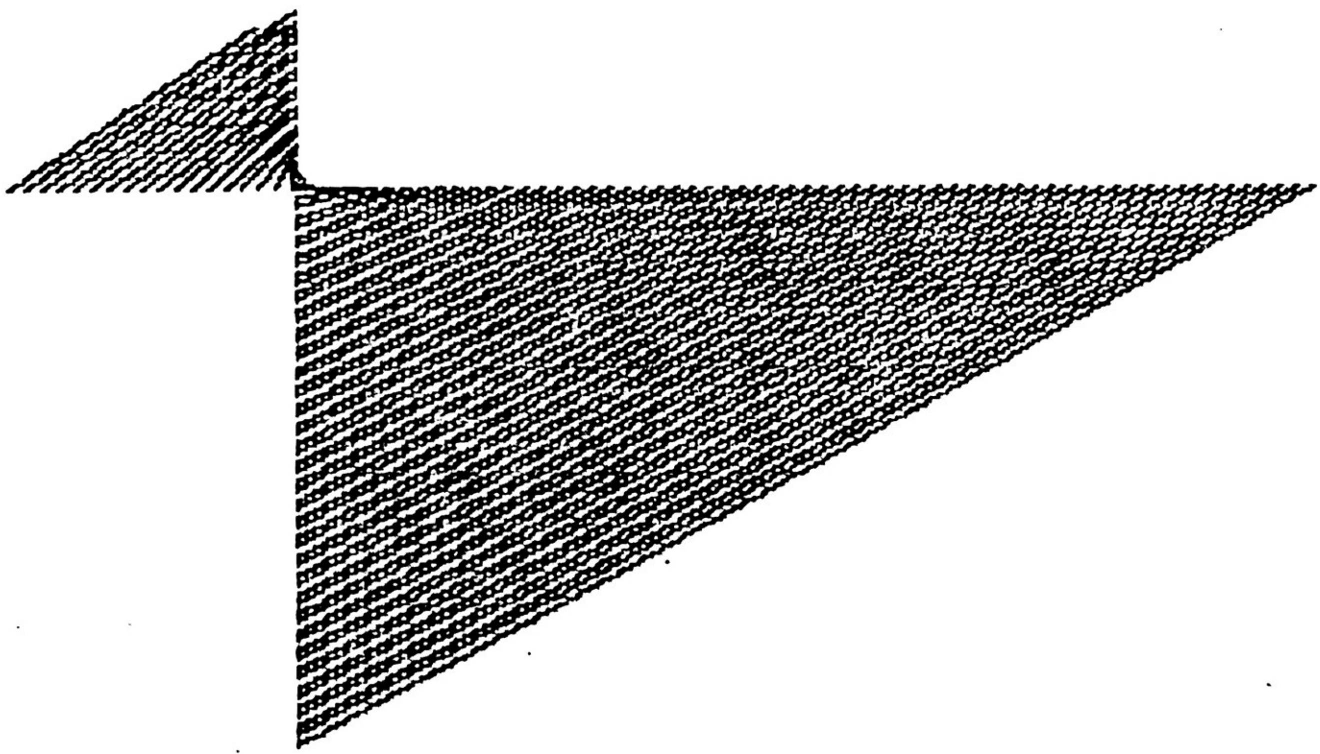
Az alprogramba

```
SYS HC, 1
```

utasítással ugrunk (logikai file-számként előzőleg 1-et választottunk). A végén ne felejtkezzünk meg a nyomtatócsatorna lezárásáról:

```
CLOSE 1
```

Ezek után próbáljuk ki a programunkat:



```
1 REM         *** P 38 ***
2 REM
3 REM
4 REM
10 REM ****************************
20 REM **                        **
30 REM ** GRAFIKAI PROGRAMCSOMAG **
40 REM **                        **
50 REM **       - D E M O -      **
60 REM **                        **
70 REM ****************************
80 REM
90 REM GEPI KODU ALPROGRAMOK
```

```
100 IN=51200:OF=51203:REM ELOKESZITES/GRAFIKA BE
110 GC=51206:SC=51209:REM GR.TAROLO TORL./SZINEK
120 PC=51212:PL=51215:REM SZINEK MOD./PONT BE
130 UP=51218:SL=51221:REM PONT KI/VONAL RAJZOLAS
140 CL=51224:GL=51227:REM VONAL TORL/GR.BETOLTES
150 GS=51230:HC=51233:REM GR.MENTESE/HARDCOPY
200 REM
210 REM PELDA
220 REM *****
230 REM
300 SYS IN: REM GRAFIKA BE
310 SYS GC: REM GRAFIKUS TAROLO TORLESE
320 SYS SC,1*16+2: REM SZINEK BEALLITASA
330 SYS PC,7*16+2 :REM UJ SZINEK
340 REM
350 REM 1. ABRA
360 REM *******
370 REM
380 FOR X=1 TO 319 STEP 4
390 SYS SL,X,50,70,X/1.6: REM VONAL
400 NEXT X
410 REM
420 FOR X=1 TO 5000:NEXT X:REM VARAKOZAS
430 GOSUB 2000:SYS GC:REM GRAFIKUS TAROLO TORL.
440 REM
450 REM 2. ABRA
460 REM *******
470 REM
480 FOR X=1 TO 319 STEP 3
490 SYS SL,X,40,50*SIN(X/30)+100,X/1.6
500 NEXT X
510 REM
520 FOR X=1 TO 5000:NEXT X:REM VARAKOZAS
530 GOSUB 2000:SYS GC
540 REM
550 REM 3. ABRA
560 REM *******
570 REM
580 FOR X=1 TO 319 STEP 2
590 SYS SL,X,40*COS(X/20)+100,50*SIN(X/30)+100,X/1.6
600 NEXT X
610 REM
620 FOR X=1 TO 5000:NEXT X
630 GOSUB 2000:SYS GC
640 REM
1000 WAIT 198,255:REM VARAKOZAS EGY BILLENTYURE
1100 SYS OF:END
2000 SYS OF:REM GRAFIKA KIKAPCSOLASA
2010 PRINT"KEREM VALASSZON !":PRINT
2020 PRINT" (1)  -  GRAFIKA BETOLTESE "
2030 PRINT" (2)  -  GRAFIKA TAROLASA  "
2040 PRINT" (3)  -  HARDCOPY          "
2050 PRINT" (4)  -  TOVABB            ":PRINT
2060 POKE 198,0  :REM BILLENTYU TORLESE
2070 WAIT 198,255:REM VARAKOZAS EGY BILLENTYURE
```

```
2080 GET A$
2090 ON VAL(A$) GOTO 2200,2300,2400,2500
2100 GOTO 2000
2200 REM
2210 REM GRAFIKA BETOLTESE
2220 REM *****************
2230 REM
2240 INPUT"FILE-NEV,KESZULEKCIM";FI$,GA
2250 SYS GL,FI$,GA:REM TOLTES
2260 SYS IN:REM GRAFIKA BE
2270 SYS SC,16*3+9
2280 FOR X=1 TO 5000:NEXT X
2290 GOTO 2000
2300 REM
2310 REM GRAFIKA TAROLASA
2320 REM ****************
2330 REM
2340 INPUT"FILE-NEV,KESZULEKCIM";FI$,GA
2350 SYS GS,FI$,GA:REM TAROLAS
2360 GOTO 2000
2400 REM
2410 REM HARDCOPY
2420 REM ********
2430 REM
2440 PRINT"KAPCSOLJA BE A NYOMTATOT ";
2445 PRINT"ES NYOMJON LE EGY BILLENTYUT !"
2450 POKE 198,0:WAIT 198,255:GET A$
2460 PRINT:PRINT"EGY KIS TURELMET KEREK !"
2470 OPEN 1,4:REM NYOMTATOCSATORNA MEGNYITASA
2480 SYS HC,1:REM HARDCOPY
2490 CLOSE 1 :REM CSATORNA ZARASA
2495 GOTO 2000
2500 REM
2510 REM TOVABB
2520 REM ******
2530 REM
2540 SYS IN:SYS SC,16*2+7:RETURN
```

Ebben a rövid kis bemutató programban néhány alkalmazási példát mutattunk be. Minden további nélkül összefűzhető saját rutinjainkkal. Sok sikert kívánunk a programozáshoz!

5. FEJEZET

Alkalmazási lehetőségek

Túljutottunk a grafikai programozás alapjainak nagy fejezetén. Megismertük a pontok, vonalak, körök, sprite-ok és a jelkészlet ábrázolásának és fejlesztésének lehetőségeit, a különböző funkciók kezelését és még sok egyebet. Itt az ideje, hogy rátérjünk ezek gyakorlati alkalmazására.

Ebben a fejezetben példákat mutatunk be azzal a céllal, hogy kedvet csináljunk a grafikai munkához. Sok trükköt is elárulunk, majd amik a saját programokban kiválóan alkalmazhatók.

Három nagy rész foglalkozik a nagyfelbontású grafikával és a sprite-okkal, majd végül bemutatunk néhány olyan dolgot, amiknek elsősorban a játékoknál lehet szerepük.

5.1 A grafika alkalmazási területei

Térjünk rögtön a tárgyra. A következőkben bemutatásra kerülő programok ugyanazokat az alprogramokat használják, mint az előző fejezetben ismertetett grafikai programcsomag. Így gyorsabban haladhatunk. Ha a programcsomagot még nem rögzítettük, akkor az előzőekben közölt BASIC-programokat is felhasználhatjuk, csupán a paramétereket kell beírni a megfelelő tárrekeszekbe és az aktuális alprogram lehívható.

A programok elején ne felejtsük el a V és GT változókba betölteni a VIC és a grafikustár kezdőcímét (V=53248; GT=8192). Ha rendelkezünk valamilyen grafikus bővítővel, akkor a SYS-utasításokat a megfelelő kiegészítő utasításokkal helyettesíthetjük. A programokat úgy állítottuk össze, hogy az ilyen módosítások könnyen elvégezhetők legyenek.

A nagyfelbontású grafika nagyon sokoldalúan alkalmazható. Nézzünk meg néhányat a kínálkozó lehetőségekből:

5.1.1 Függvényábrázolás

A nagyfelbontású grafika alkalmazásának egyik leglátványosabb példája a függvények grafikus ábrázolása. Ezzel a 4.2.2.1 fejezetben a szinusz- és koszinuszgörbe ábrázolásakor már találkoztunk. Egyébként is szívesen alkalmazzák példaprogramokban, mert látványos és nem okoz problémát a képernyőhatárok túllépése. Képek előállításánál ez utóbbi okozza a legtöbb problémát.

Mielőtt magával a programmal kezdenénk foglalkozni, ismételjük át röviden a témához szükséges matematikai ismereteket.

A függvény fogalma azt jelenti, hogy egy mennyiséget egy másikra elemenként leképezünk. Az első mennyiség elemeit az algebrában x-szel, a másodikét y-nal jelöljük. A köztük levő kapcsolatot valamilyen matematikai képlettel, ill. egyenlettel fejezzük ki. Ezt láttuk a 4.2.2.2 fejezetben, ahol az egyenes egyenletével dolgoztunk.

$$y = f(x)$$

Ebből x bármely értékére kiszámítható a hozzá rendelt y érték.

Az x és y értékek közti összefüggést egy koordináta-rendszerben, grafikusan is ábrázolhatjuk. A koordináta-rendszer vízszintes tengelyén (abszciszsza) jelöljük az x, a függőleges tengelyén (ordináta) pedig az y értéket. A függvény ábrázolása úgy történik, hogy a tetszőlegesen választott x értékből kiszámítjuk a hozzá rendelt y-t, majd mindkettőt a megfelelő tengelyre felmérve a metszéspontba egy pontot rajzolunk. Megfelelően sok pont kiszámítása és berajzolása után ezeket összekötve megkapjuk a függvény grafikus képét.

Az x és y értékek a függvény pontjainak koordinátái. Programban ezek kiszámítására egy FOR...NEXT-ciklust szervezhetünk, ahol x-et a megfelelő léptékkel ciklusonként növeljük (vagy csökkentjük), majd az így kapott értékeket a képletbe behelyettesítve kiszámítjuk az y-okat. Az így, most már x és y koordinátákkal, meghatározott pontokat valamilyen eljárással elhelyezzük a képernyőn. A 4. fejezetben ezt az elvet alkalmaztuk az egyenes és a kör megrajzolásakor.

A következő programban még egyszer bemutatjuk az eljárást:

```
1 REM *** P 39 ***
2 REM
70 REM GEPI KODU ALPROGRAMOK
100 IN=51200:OF=51203:REM ELOKESZITES/GRAFIKA KI
110 GC=51206:SC=51209:REM GR.TAROLO TORL./SZINEK
120 PC=51212:PL=51215:REM SZINEK MOD./PONT BE
130 UP=51218:SL=51221:REM PONT KI/VONAL RAJZOLAS
140 CL=51224:GL=51227:REM VONAL TORL./GR.LOAD
150 GS=51230:HC=51233:REM GR.SAVE/HARDCOPY
160 REM
170 REM **************
180 REM
200 SYS IN:SYS GC:SYS SC,16*1+6
210 FOR X=0 TO 319
220 Y = X↑2:REM FUGGVENYERTEK SZAMITASA
230 IF Y>199 THEN WAIT 198,255:SYS OF:END
235 REM TARTOMANY TULLEPES
240 SYS PL,X,Y:REM PONT RAJZOLASA
250 NEXT X
```

A 230-as sorban ellenőrizzük, hogy y értéke nem lépte-e túl a képernyőhatárokat. Ha nem, a program a rajzoló alprogramban folytatódik és a pont a képernyőre kerül. Ha igen, a program egy billentyű lenyomására vár, majd a befejező alprogramba ugrik. Erre azért van szükség, hogy elkerüljük az ilyen esetben egyébként bekövetkező

ILLEGAL QUANTITY ERROR (=nem megengedett mennyiség)

hibaüzenetet.

A képernyőn megjelenő kép nem felel meg a várakozásunknak. Először is a két parabolaág közül csak az egyik kerül a képernyőre, ráadásul az is fordítva. Ezenkívül túl kevés pontot számítottunk ki, így nem kaptunk igazán szép görbét. Ahhoz, hogy ezeket a hibákat kijavíthassuk, a következőket kell tudnunk: számítógépünk nem teszi lehetővé, hogy x-nek és y-nak negatív vagy tört értéket adjunk, valamint x értéke legfeljebb 319, y-é pedig legfeljebb 199 lehet. Némelyik függvény viszont épp ebben a meg nem engedett tartományban érdekes. Ilyenek az egyszerű szögfüggvények is, mint pl. az

$y = \sin(x)$, vagy az

$y = \cos(x)$.

Ezek -1 és $+1$ közötti értékeket vehetnek fel. Periódusuk 2π, így kb. hatszor férnek el a képernyőablak területén. Szerencsére léteznek olyan módszerek, amelyek ebben a kényszerhelyzetben segítségünkre lehetnek. Ezek a

- kicsinyítés, nagyítás,
- eltolás,
- torzítás.

a) *Kicsinyítés, nagyítás*

Kicsinyítés és nagyítás alatt azt értjük, hogy az f(x) függvényt egy tetszőleges számmal megszorozzuk. Ezáltal a kiszámított y értékek kisebbek vagy nagyobbak lesznek, mint a normál függvényértékek és a görbe nagysága megváltozik. Az eljárás kétféleképpen is megvalósítható.

1. Az y értékeit az eredeti függvény szerint számítjuk ki és az így kapott értéket szorozzuk a kiválasztott kicsinyítő vagy nagyító tényezővel (f). A kiválasztás az alábbiak szerint történik:

   $0<f<1$ kicsinyítés,
   $a>1$ nagyítás.

   Ha $f<0$, a képernyőn a függvény vízszintes tengelyre vonatkozó tükörképe jelenik meg.

   Az eljárást a következő BASIC-rutinnal szemléltetjük:

```
1 REM *** P 40 ***
2 REM
3 REM
4 REM
100 IN = 51200 : GC = 51206 : SC=51209
102 OF = 51203 : PL = 51215
105 SYS IN : SYS GC : SYS SC,16*0+1
110 F = 30 :REM NAGYITASI TENYEZO
120 FOR X=0 TO 10
130 Y=SIN(X):REM ERTEKEK KISZAMITASA
140 Y=F*Y+100:X=F*X:REM NAGYITAS
150 IF Y>199 OR X>319 THEN WAIT 198,255:SYS OF:END
160 SYS PL,X,Y:REM PONTOK ELHELYEZESE
170 NEXT X
```

   A program csak a grafikus programcsomaggal (P37) együtt futtatható és előzőleg definiálnunk kell a változókat. A képlet negatív y értékeket szolgáltat.

2. A második lehetőség az, hogy eleve a kicsinyített vagy nagyított y értéket számítjuk ki az

$$y = f*f(x)$$

össszefüggés szerint, x értékeit már a FOR...NEXT-ciklusban f-fel szorozva határozzuk meg. Programban ez a következőképpen alakul:

```
1 REM *** P 41 ***
2 REM
3 REM
4 REM
100 IN = 51200 : GC = 51206 : SC=51209
102 OF = 51203 : PL = 51215
105 SYS IN : SYS GC : SYS SC,16*0+1
110 F = 30 :REM NAGYITASI TENYEZO
120 FOR X=0*F TO 10*F
130 Y=F*SIN(X/F)+100:REM ERTEKEK KISZAMITASA
140 IF Y>199 OR X>319 THEN WAIT 198,255:SYS OF:END
150 SYS PL,X,Y:REM PONTOK ELHELYEZESE
160 NEXT X
```

Ennek a módszernek három előnye is van. Először: rövidebb és gyorsabb. Másodszor: az f*x szorzat nem lépheti túl a megengedett értéket, mivel azt a FOR...NEXT-ciklusban lehatároltuk. Harmadszor: a fenti 11 helyett összesen 301 pontot határozhatunk meg és így sokkal szebb görbét kapunk. Már csak egy gondunk van: y továbbra is negatív. Ezt a következő eljárással küszöbölhetjük ki:

b) *Eltolás*

A koordináta-rendszeren belül bármelyik grafikonunkat bármelyik irányba eltolhatjuk. Az előbbi szinuszgörbénket tehát minden további nélkül eltolhatjuk úgy, hogy y csak pozitív értékeket vegyen fel. A P39-es programmal ábrázolt másodfokú függvényünk képét is eltolhatjuk valamilyen értékkel jobbra, hogy a bal oldali ága is láthatóvá váljék.

Az eltolást úgy hajtjuk végre, hogy a koordinátaértékekhez hozzáadunk valamilyen összeget. Nem fontos, hogy az x és y irányú eltolás mértéke azonos legyen, sőt az egyik akár 0 is lehet. A kicsinyítés, ill. nagyítás esetében erre azért volt szükség, mert amennyiben a koordinátaértékeket eltérő mértékben változtatjuk meg, megváltozik az x és y értékek hozzárendelési viszonya és ez a görbe alakjának torzulásához vezet. Az eltolás esetében ez nem áll fenn.

Az y értékét b-vel növelve (ill., ha $b<0$, csökkentve) a görbét y irányban

felfelé (ill. lefelé) toljuk el. Ugyanez, ha x-et a-val növeljük (ill. csökkentjük), a görbe jobbra (ill. balra) történő eltolását jelenti. Ebben az esetben is két lehetőségünk van:

1. Kiszámítjuk a függvényértékeket, majd ehhez hozzáadjuk az a és b állandókat. Példaként cseréljük ki a P40-es program 130-as és 140-es sorát az alábbi két sorral:

```
130 Y = SIN (X)
140 Y = Y+B:X = X+A
```

2. Az eltolási tényezőket beépítjük a képletbe. Ebben az esetben a 120-as és 130-as sorokat kell kicserélnünk:

```
120 FOR X=1 TO 10
130 Y = SIN (X-A)+B
```

Az értéket most kivontuk, ezzel a görbét balra toltuk el.

A másodfokú függvényünk programját (P39) a következőképpen módosíthatjuk:

```
220 Y = (X-13) ↑ 2+5
```

ahol a= −13 és b=5.

Így már egészen más eredményt kapunk. Ha a és b értékét túl nagyra választjuk, a görbe képe nem jelenik meg a képernyőn, mert a 230-as sorban előírt ellenőrzés eredményeként, a képernyőhatárokon kívülre kerülő értékek esetén, a program befejeződik.

Erre később még visszatérünk.

c) *Torzítás*

A torzítás egyszerűen a méretváltoztatás általános esete. Ilyenkor az x-et és y-t eltérő értékekkel szorozzuk meg, amiket a továbbiakban f1-gyel és f2-vel jelölünk. Ebben az esetben megváltoznak a görbe arányai, tehát zsugorítás ($0<f1/f2<1$) vagy nyújtás ($f1/f2>1$) történik. A legtöbb görbe valójában csak így ábrázolható. Vegyük elő ismét a másodfokú görbénket és módosítsuk a program 220-as sorát a következőképpen:

220 Y = 0.01*((X–139)/1) ↑ 2+5

Egyszerre alkalmazzuk a torzítást és az eltolást, hogy helyes eredményt kapjunk. f1 = 1, f2 = 0.01, a = –139, b = 5.

Valamit javult a helyzet, de a parabolánk még mindig fejen áll és a paraméterek változtatásakor időnként még előfordul az

ILLEGAL QUANTITY ERROR (=nem megengedett érték hiba)

hibaüzenet. Mi lehet ennek az oka? Nos, egyszerűen az, hogy maga a koordináta-rendszer áll a fején. Középpontja ugyanis nem a bal alsó, hanem a bal felső sarokban van, és y értékei felülről lefelé nőnek. Kézenfekvő a megoldás: fordítsuk meg a görbénket úgy, hogy az y értékek előjelét megfordítjuk. Mivel gyakran előfordul, hogy az y csak negatív értékeket vesz fel, el is kell tolnunk a görbét lefelé (tulajdonképpen felfelé) pl. 195-tel. Módosítsuk ennek megfelelően a program 220-as sorát:

220 Y = –0.01*((X–139)/1) ↑ 2+195

Elgondolásunk helyességét az eredmény igazolja.

Térjünk rá most az egyre gyakoribbá váló hibaüzenetekre. Ennek oka a nem tökéletes tartomány-ellenőrzésben keresendő. Egyrészt egyáltalán nem ellenőrizzük, hogy y negatív-e, másrészt y>199 esetén azonnal befejeződik a program. Írjuk át a P39-es programot a 210-es sortól a következőképpen, és indítsuk el a programot.

```
1 REM *** P 42 ***
2 REM
3 REM
4 REM
100 IN = 51200 : OF = 51203 :GC = 51206
110 SC = 51209 : PL = 51215
120 SYS IN : SYS GC : SYS SC,16*15+4
210 F1= 1 : F2=0.1 : A=160 : B=150
220 FOR X=0 TO 319
230 Y=-F2*((X-A)/F1)↑2+B
240 IF Y>199 OR Y<0 THEN NEXT X:GOTO 270
250 SYS PL,X,Y:REM PONTOK ELHELYEZESE
260 NEXT X
270 WAIT 198,255:SYS OF:END
```

Mielőtt a számítógép rajzolni kezdene, egy csomó értéket kiszámít. Ez egy kissé hosszadalmas művelet, ezért legyünk türelemmel, ne szakítsuk meg a programot. A görbe most a képernyő közepére került (a = 160, b = 100). Szemléletesebbé tehetjük a grafikont, ha ide berajzoljuk a koordinátatengelyeket. A koordináta-rendszer középpontja a képernyő 160, 100 koordinátájú pontja.

A BASIC egyik különleges képessége, hogy az egyszer már definiált függvényre a programban bárhol hivatkozhatunk anélkül, hogy azt újra és újra le kellene írnunk. Ennek különösen bonyolult összefüggések esetén van óriási jelentősége. A megfelelő BASIC-utasítás a DEF FN. Ismerkedjünk meg az alkalmazásával. Módosítsuk a programunkat a következők szerint:

```
215 DEF FB F(X) = -F2*((X-A)/F1) ↑ 2+B
230 Y = FN F(X)
```

Ha csupán az alapfüggvényt akarjuk tárolni:

```
215 DEF FN F(X) = X ↑ 2
230 Y = -F2*FN F((X-A)/F1+B
```

A változók mindenkori értékeit behelyettesíthetjük a képletbe. Ezzel a módszerrel egyszerűen módosíthatjuk a függvényeket anélkül, hogy a ciklusba be kellene avatkoznunk, vagy a függvényt keresnünk kellene.

A következő programmal egy kicsit tovább lépünk. Segítségével bármilyen függvény képe kirajzolható. Egy INPUT-lekérdezéssel beírhatjuk a kiválasztott összefüggést, amiből a program képezi a megfelelő függvényutasítást. Nem fontos a rutint tökéletesen érteni, de figyeljük meg a DEF FN-utasítás alkalmazását. A függvényt torzítás és eltolás nélkül kell beírni. Ezekre majd a főprogramban kerül sor.

Figyelem! A program bevitelekor, különösen a 350-es sorig, ügyeljünk a pontos másolásra. Ebben az esetben a formai jegyeknek is jelentőségük van, beleértve a szóközöket is. Indítás előtt feltétlenül rögzítsük a programot!

```
1 REM *** P 43 ***
2 REM
3 REM
4 REM
100 REM ************************
110 REM **                    **
120 REM ** FUGGVENY ABRAZOLAS **
130 REM **                    **
140 REM ************************
150 REM
160 REM
170 REM DEF FN - ALPROGRAM
180 REM ******************
190 REM
200 RESTORE:DIM A$(25):PRINT CHR$(147)
210 PRINT"KEREK EGY TETSZOLEGES FUGGVENYT !":PRINT:PRINT:PRINT
220 GOSUB 500:REM BEVITELI ALPR.
230 AD=2736-1:REM POKE KEZDOCIM
240 FOR X=1 TO 25:READ A$(X):NEXT X:REM ADATOK BE
250 FOR X= 1 TO LEN(A$):AD=AD+1
260 G=X:FOR Y=1 TO 25:B$=A$(Y):FOR Z=1 TO LEN(B$):C$=MID$(A$,X,1)
270 IF C$=" " THEN AD=AD-1:NEXT X:GOTO 310
280 IF C$=MID$(B$,Z,1) THEN X=X+1:NEXT Z:POKE AD,Y+169:X=X-1:NEXT X:GOTO 310
290 X=G:NEXT Y
300 POKE AD,ASC(C$):NEXT X
310 POKE AD+1,58:POKE AD+2,143
340 REM HELY BIZTOSITAS
350 DEFFNF(X)=X*X:REM000000000000000000000000000000000000000000000000000000000
000000
400 REM 25 FUGGVENY:
410 DATA +,-,*,/,↑,AND,OR,>,=,<,SGN,INT,ABS,USR,FRE,POS,SQR,RND,LOG,EXP
420 DATA COS,SIN,TAN,ATN,PEEK
430 GOTO 600:REM TOVABB
500 INPUT"F(X)=";A$:RETURN
540 REM
550 REM **************************
560 REM ** GRAFIKAI ALPROGRAMOK **
570 REM **************************
580 REM
600 IN=51200:OF=51203:GC=51206
610 SC=51209:PC=51212:PL=51215
630 UP=51218:SL=51221:CL=51224
640 GL=51227:GS=51230:HC=51233
655 POKE 2,0:REM TORLES FLAG
660 REM
670 REM ***************
680 REM ** FOPROGRAM **
690 REM ***************
700 REM
710 PRINT:PRINT"KEREM A KOVETKEZO PARAMETEREKET !"
720 INPUT"F1 (X-TORZ.TENYEZO)";F1
730 INPUT"F2 (Y-TORZ.TENYEZO)";F2
740 INPUT"A  (X-ELTOLAS      )";A
750 INPUT"B  (Y-ELTOLAS      )";B
1000 SYS IN:SYS SC,16*1+5
```

```
1005 IF PEEK(2)=0 THEN SYS GC:REM GRAFIKA TORL.
1007 POKE 2,0
1010 REM
1020 REM TENGELYEK MEGRAJZOLASA
1030 REM ************************
1040 IF A>=0 AND A<320 THEN SYS SL,A,0,A,199
1045 REM X-TENGELY
1050 IF B>=0 AND B<200 THEN SYS SL,0,B,319,B
1055 REM Y-TENGELY
1060 REM
1070 REM RAJZOLAS
1080 REM ********
1210 FL%=1:REM ERTEKHATAR JELZO
1220 FOR X=1 TO 319
1230 Y=-F2*FNF((X-A)/F1)+B
1240 IF Y<0 OR Y>199 THEN FL%=1:NEXT X:GOTO 1280
1250 IF FL%=0 THEN SYS SL,X1,Y1,X,Y
1260 FL%=0
1270 X1=X:Y1=Y:NEXT X:REM UTOLSO KOORD.
1280 POKE 198,0:WAIT 198,255:GET A$:SYS OF
1285 REM GARAFIKA KI
1290 REM MENU
1295 REM ****
1300 PRINT"KEREM VALASSZON !":PRINT:PRINT
1310 PRINT" (1) UJ PARAMETEREK"
1320 PRINT" (2) UJ FUGGVENY"
1330 PRINT" (3) GRAFIKA MARAD"
1340 PRINT" (4) GRAFIKA TAROLASA"
1350 PRINT" (5) GRAFIKA BETOLTESE"
1360 PRINT" (6) HARDCOPY"
1390 PRINT" (7) BEFEJEZES"
1450 WAIT 198,1:GET A$:PRINT A$
1460 ON VAL(A$) GOTO 700,1480,1490,1510,1600,1650,1500
1470 GOTO 1450
1480 RUN:REM UJ FUGGVENY
1490 POKE 2,1:PRINT"RENDBEN":PRINT:GOTO 1450
1500 END:REM BEFEJEZES
1510 REM
1520 REM GARAFIKA TAROLASA
1530 REM ******************
1540 INPUT"FILE-NEV,KESZULEKCIM";FI$,GA
1550 SYS GS,FI$,GA:REM TAROLAS
1560 PRINT"RENDBEN !":GOTO 1450
1570 REM
1590 REM
1600 INPUT"FILE-NEV,KESZULEKCIM";FI$,GA
1610 SYS GL,FI$,GA:REM BETOLTES
1630 REM
1640 REM HARDCOPY
1645 REM ********
1650 PRINT"KAPCSOLJA BE A NYOMTATOT ES";
1655 PRINT" NYOMJON LE EGY BILLENTYUT !"
1660 OPEN 1,4:SYS HC,1:CLOSE1:REM HARDCOPY
1670 PRINT"RENDBEN !":GOTO 1450
```

A függvénygörbe megjelenésére most is várni kell egy kicsit, legyünk türelemmel!

A program a P37-es programmal együtt működik. Más grafikai rutinok alkalmazása esetén a programot az 5. fejezetben leírtak szerint módosítani kell.

Néhány szó a program működéséről:

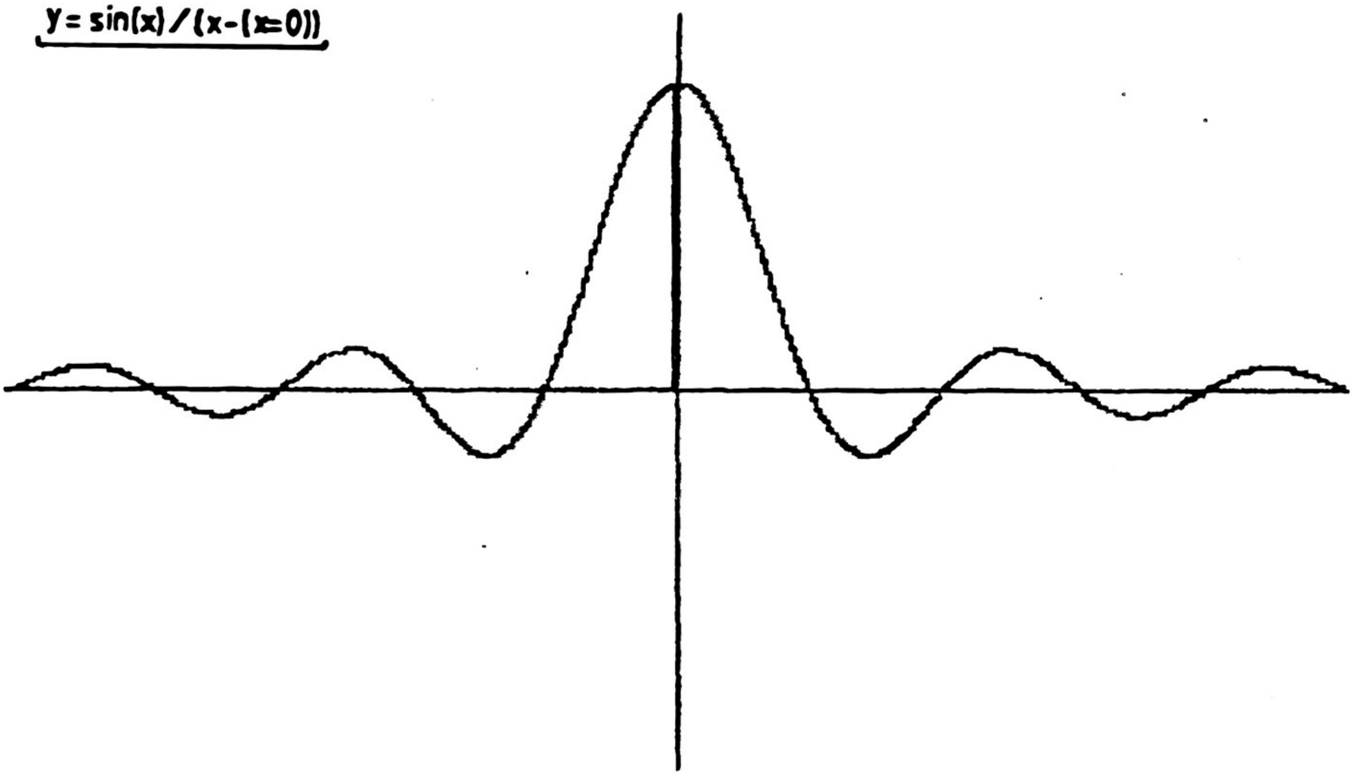
A kiválasztott összefüggést az 500-as sorban levő INPUT-utasítás kéri be. Ezt GET-tel is helyettesíthetjük. Azért került a program ilyen távoli részére, mert előtte a 350-es sorig a programban semmiféle változtatás nem végezhető. A beírt összefüggés az A$ változóba kerül, majd a 350-es sorig terjedő átalkító rutinban egy DEF FN-sor lesz belőle. Ez úgy történik, hogy a rutin a szükséges byte-okat közvetlenül a BASIC-tárba írja (POKE) és ezáltal módosítja a 350-es sort. Az átírás kezdőcímét a 230-as sorban adtuk meg. Aki kiismerte a programot, megérti, miért mondtuk azt, hogy az első része (a 350-es sorig) nem módosítható, ill. ezek után akár már ez is megtehető. A 410/420-as sorokban felsorolt műveletek bármelyike szerepelhet az összefüggésünkben, betartva a BASIC-ban előírt írásmódot.

A belső rutinok kezdőcímeinek megadását a torzítási és eltolási tényezők (f1/f2 és a/b) bevitele követi. Itt alkalmazhatjuk az előbb tanultakat. Néhány tájékoztató érték: f1 és f2 értékét a legtöbb függvény esetében 1-nél nagyobbra kell választanunk. Szögfüggvényeknél például a 10–70 tartományt ajánljuk. Mint tudjuk, f1 a vízszintes, f2 a függőleges irányú nyújtásért felelős. Az eltolás értékeiként a = 160-at és b = 100-at javaslunk. Ebben az esetben ugyanis a görbe a képernyő középpontjába kerül és így a legegyszerűbb az áttekintése. Ha csak a pozitív tartományra vagyunk kíváncsiak, a = 0-t válasszuk.

A bekapcsolás és a grafikustár esetleges törlése (törlésjelzőtől függően – 1007-es sor) után megrajzoljuk a koordinátatengelyeket (1040-es sor). Tetszés szerint megrajzolható a tengelyek beosztása is. A görbe ábrázolása csak a 1210-es sorban kezdődik. Itt találkozunk az előzőekben megismert programrésszel, ami csak abban tér el az eredetitől, hogy a kiszámított és felrajzolt pontokat egyenesekkel össze is kötjük. Problémát csak az jelent, hogy ha a görbe érvénytelen tartományból indul, az utolsó ponttól a pontosan kiszámított pontig nem húzható egyenes. Erről az FL%-jelző (integer-változó) gondoskodik.

A görbe felrajzolása után egy tetszőleges billentyű lenyomásával visszatér-

hetünk a főmenübe. Választhatunk új függvényt, módosíthatjuk a régit (összehasonlítás céljából), ábrázolhatjuk a régi és új függvényt együtt, tárolhatjuk vagy betölthetjük az elkészült képet, készíthetünk hardcopyt, vagy egyszerűen befejezhetjük a programot.



Még valamit az összefüggés beviteléről: ügyeljünk a helyes írásmódra, mert könnyen SYNTAX ERROR hibaüzenetet idézhetünk elő. Vigyázzunk arra, hogy ne hozzunk létre matematikailag helytelen kifejezéseket, pl. 0 a nevezőben vagy a logaritmusban, negatív érték a logaritmusban vagy a gyökvonásban... stb. Az előző eset elkerülése érdekében az alábbi trükköt alkalmazhatjuk:

1/X	helyett	1/(X–(X=0))	vagy
LOG(X)	helyett	LOG(ABS(X–(X=0))	

Ha X=0, a Commodore BASIC-ben az X=0 kifejezés értéke –1, egyébként 0. Hasonlóan alkalmazhatók a relációs jelek (<, >) is.

Az ABS-függvényt a negatív értékek kiküszöbölésére alkalmazzuk. Pl.:

SQR(ABS(X))

Ha valami hiba merül fel, a program RUN 350 utasítással újraindítható. A bevitt függvény nem vész el.

5.1.2.3 Háromdimenziós grafika

Aligha van olyan számítógépes reklám, amelyik ne használná ki a háromdimenziós grafika lehetőségeit. Nem csoda, hisz az így készült rajzok látványosak, esztétikusak és éppen ezért kiválóan alkalmasak propagandacélokra. A tervezők egyre gyakrabban foglalkoznak háromdimenziós függvények ábrázolásával. A magánember számára is rengeteg érdekességet tartogat ez a téma, de egyben némi előképzettséget is igényel.

A matematikai algoritmusok levezetése nem egyszerű feladat, ezért bemutatunk néhány eljárást, amelyek az 5.1.1 fejezetre épülnek.

5.1.2.4 Párhuzamos leképezés

Alapvetően kétféle ábrázolási módot különböztetünk meg:

- párhuzamos leképezés,
- középpontos (centrális) leképezés.

Ebben a fejezetben az első és egyszerűbb módszerrel foglalkozunk. Ez egyben előkészíti a másodikat, amivel a valóság már elég jól megközelíthető.

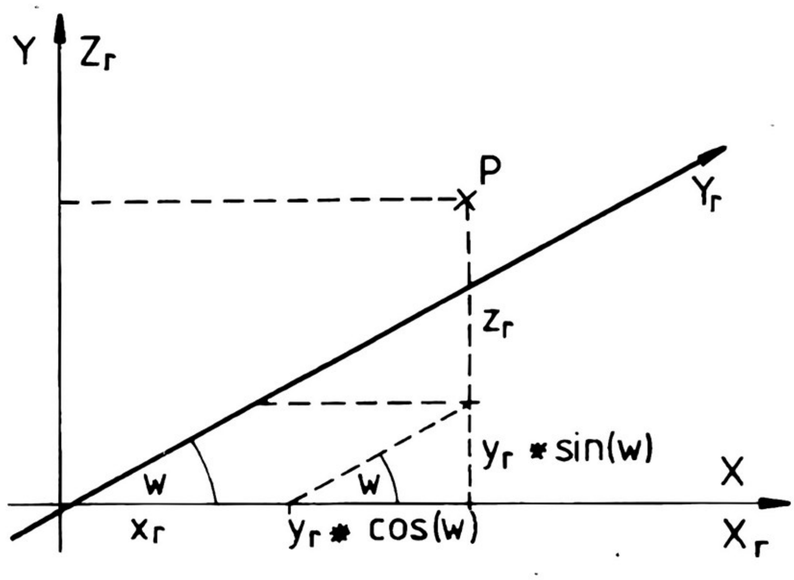
Első teendőnk, hogy továbbfejlesszük kétdimenziós koordináta-rendszerünket. A háromdimenziós koordináta-rendszerben három koordinátatengelyünk van, amelyeket xr-rel, yr-rel és zr-rel jelölünk. Az r jelenti a térbeliséget. Így tehát egy térbeli tárgy (pl. egy ház) minden pontja három koordinátával jellemezhető. Mivel mindezt egy síkbeli koordináta-rendszerben (a képernyő síkja) kell ábrázolnunk, a térkoordinátákat síkkoordinátákba kell átszámítanunk. Ehhez először meg kell határoznunk a térbeli koordináta-rendszer elhelyezkedését.

Az xr–zr tengelyek által kijelölt sík megegyezik a képernyő síkjával. Ez a két tengely egymásra merőleges. Az yr tengely a valóságban merőleges az xr–zr tengelyekkel meghatározott síkra, ami a képernyőn úgy jelenik meg, hogy az xr és yr tengelyek egymással w szöget zárnak be.

Az ábrából a koordináták átszámításához a következő összefüggéseket kapjuk:

x = xr+yr*cos (w),

y = zr+yr*sin (w).



A w szög határozza meg a perspektívát, vagyis azt a szöget, amely alatt a tárgyat látjuk.

Ha a rajzot el akarjuk tolni, az 5.1 fejezet szerint kell eljárnunk, azaz az xr, yr és zr koordinátákhoz rendre hozzá kell adnunk az a, b és c értékeket (eltolási tényezők):

x = a+xr+(yr+c)*cos (w),

y = b+zr+(yr+c)*sin (w).

Ha az ábrát csak a képernyő síkjában akarjuk eltolni, C = 0.

Előfordulhat, hogy a kép nem látszik, vagy mert túl kicsi, vagy mert egyes részei kívül esnek a képernyőhatárokon. Ebben az esetben itt is alkalmazhatjuk a torzítást és a kicsinyítést, ill. nagyítást. Ekkor megváltozik az ábra szélessége, hosszúsága és magassága. A torzítási tényezőket f1-gyel, f2-vel és f3-mal jelöljük, amelyekkel az xr, yr és zr koordinátákat közvetlenül módosítjuk:

x = f1*(a+xr)+f3*(yr+c)*cos (w),

y = f2*(b+zr)+f3*(yr+c)*sin (w).

Ha az ábrát nem akarjuk torzítani, csak kicsinyíteni vagy nagyítani, akkor a három tényezőnek egyenlőnek kell lenni. A tényezők értéke a következők szerint módosítja az eredeti méreteket:

f1/f2/f3 > 1 – nagyítás,

0 < f1/f2/f3 < 1 – kicsinyítés.

Ahhoz, hogy a rajz a képernyőre kerüljön, be kell vezetnünk a síkbeli koordináta-rendszerre vonatkozó v1 és v2 eltolási tényezőt. Az előző ±x irányú, az utóbbi ±y irányú eltolást eredményez. Most is találkozni fogunk azzal a jelenséggel, hogy a rajzunk fejen áll. Az y koordináták előjelét tehát megint meg kell fordítanunk.

x = f1*(a+xr)+f3*(yr+C)*cos (w)+v1,

y = f2*(b+zr)–f3*(yr+C)*sin (w)+v2.

A következő programmal egy ház térbeli képét fogjuk megrajzolni. A sarokpontok koordinátáit DATA-sorokban adtuk meg (1110–1140). A térkoordinátákat átszámítjuk síkkoordinátákká és az így kijelölt pontokat egyenesekkel összekötjük. Azt, hogy melyik pontokat kell egymással összekötni, szintén DATA-sorokban rögzítjük (2100–2150). A pontokat a sorok szerint 1-től kezdődően megszámozzuk és az összekötés sorrendjében beírjuk a DATA-ba. A két első adat (10 és 17) a berajzolandó pontok, ill. vonalak számát jelenti (1100-as és 2100-as sor). Ezzel a módszerrel bármilyen alakzat adatait megadhatjuk, akár saját számítógépünkét is. Precízségünktől függ, hogy milyen hűen látjuk viszont a képernyőn.

Felmerülhet a kérdés, hogy miért kell mind a három koordinátát megadni. Erre azért van szükség, mert ahhoz, hogy a tárgyat különböző beállításban ábrázolhassuk, kicsinyíthessük, nagyíthassuk, vagy eltolhassuk, a térbeli elhelyezkedését jellemző koordinátákat együttesen kell módosítanunk. Kölcsönhatásuk elég bonyolult. Az f1, f2 és f3 tényezőkkel például a koordinátatengelyek egységeit is meghatározzuk, ezért megváltoztatásukkal a tárgy képe előre és hátra fog mozogni. Ezt a képlet megfelelő átalakításával kerülhetjük el:

x = f1*a+xr+f3*yr*cos (w)+c+v1,

y = –f2*b–zr–f3*yr*sin (w)–c+v2.

Igaz, hogy a matematikai átalakítást nem teljesen szabályosan hajtottuk végre, de csak így értük el a célunkat. A torzítási tényezők nem befolyásolják többé a távolságot. A torzítás csak az ábrára vonatkozik, a koordinátarendszerre nem. Problémát most már csak az okozna, ha a tengelybeosztást is fel akarnánk rajzolni.

Érdekes dologgal találkozunk a 600–630-as sorokban. Itt történik az xr és zr tengelyek megrajzolása. Ahhoz, hogy a rá merőleges yr tengelyt is

ábrázolhassunk, egyszerűen meg kell keresni azt a pontot, amelynek y koordinátája 0 (ill. a w szöget is figyelembe véve 199), és ezt össze kell kötnünk a koordináta-rendszer középpontjával.

Minden rajzolás előtt ellenőrizni kell azt, hogy a kialakult értékek a megengedett tartományban vannak-e. Egy rajzprogram akkor lenne igazán tökéletes, ha a képernyőn kívülre eső ábrarészek figyelembevételével meg tudná rajzolni a látható részeket. Ez a háromdimenziós grafika legnehezebb és legunalmasabb fejezete. Hasonlóan bonyolult, mint a háromdimenziós ábrák takart vonalrészeinek meghatározása, amire komoly fejlesztőintézetek próbálnak minél jobb megoldásokat kidolgozni.

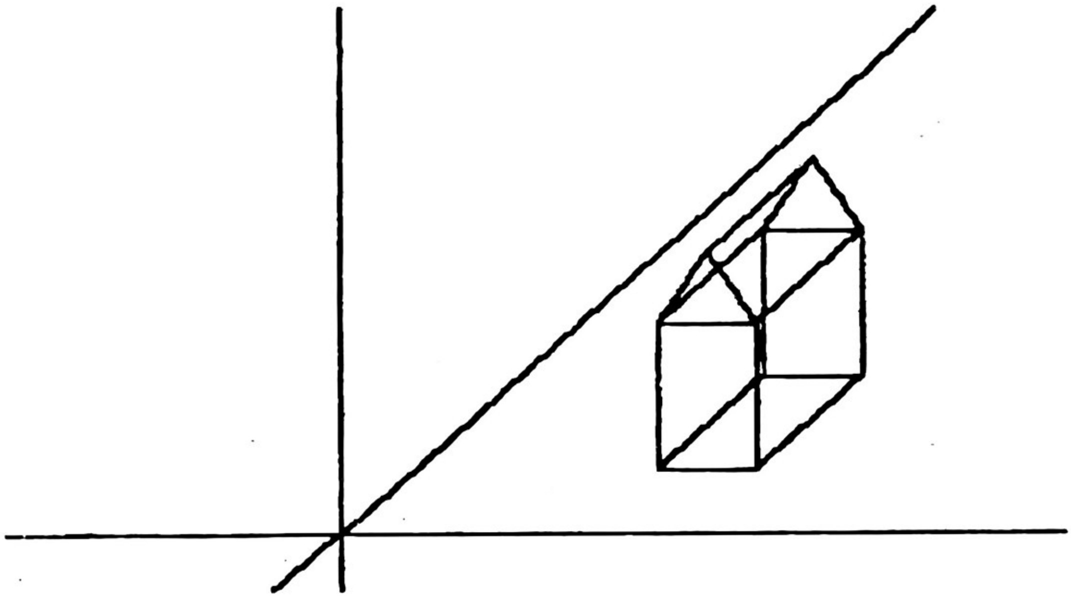
A mi programunk ilyet nem tud, a takart vonalakat is megrajzolja. Később bemutatunk majd egy eljárást, amivel valami hasonlót érthetünk el és alkalmazható a háromdimenziós függvényeknél.

A programot továbbfejleszthetjük. Lemezre rögzíthetjük a DATA-sorok adatait, amiket később bármikor betölthetünk. Természetesen ilyenkor nem a DATA-sorokat használjuk, hanem az adatokat tömbben, szekvenciális fileként tároljuk.

Több ábra adatait rögzítve menüből választhatjuk ki a megfelelőt.

```
1 REM *** P 44 ***
2 REM
3 REM
4 REM
100 REM ***************************
110 REM **                       **
120 REM ** HAROMDIMENZIOS ABRAK **
130 REM **                       **
140 REM ***************************
150 REM
200 REM
210 REM ***************************
220 REM ** GRAFIKAI ALPROGRAMOK **
230 REM ***************************
240 REM
250 IN=51200:OF=51203:GC=51206
260 SC=51209:PC=51212:PL=51215
280 UP=51218:SL=51221:CL=51224
290 GL=51227:GS=51230:HC=51233
400 REM
410 REM ***********
420 REM PARAMETEREK
430 REM ***********
440 REM
450 F1=5:F2=5:F3=3:REM TORZITAS
460 AR=15:BR=0:CR=10
```

```
470 W=3.1415/4:REM LATOSZOG (RAD)
480 SI=SIN(W):CO=COS(W):REM ALLANDOK
490 V1=100:V2=180
500 REM
510 REM ****************
520 REM ** ATSZAMITAS **
530 REM ****************
540 REM
550 SYS IN:SYS GC:SYS SC,16*7+8
560 FG=0:IF V1<0 OR V1>319 THEN FG=1:GOTO 580
570 SYS SL,V1,0,V1,199:REM Z-KOORD.
580 IF V2<0 OR V2>200 THEN FG=1:GOTO 600
590 SYS SL,0,V2,319,V2:REM X-KOORD.
600 Z2=V1-(199-V2)/SI*CO:Z1=V1+V2/SI*CO
610 IF FG=0 AND Z1>=0 AND Z1<320 THEN SYS SL,V1,V2,Z1,0
615 REM Y-TENGELY FEL
620 IF FG=0 AND Z2>=0 AND Z2<320 THEN SYS SL,V1,V2,Z2,199
630 READ AP:DIM X%(AP),Y%(AP):REM PONTOK SZAMA
640 FOR ZA=1 TO AP
650 READ XR,ZR,YR
660 X%(ZA)= F1*(XR+AR)+F3*(YR+CR)*CO+V1
670 Y%(ZA)=-F2*(ZR+BR)-F3*(YR+CR)*SI+V2
680 NEXT ZA:REM KOVETKEZO PONT
700 REM
710 REM ***********
720 REM ** VONAL **
730 REM ***********
740 REM
750 READ AL:REM VONALAK SZAMA
760 FOR ZA=1 TO AL
770 READ P1,P2
775 IF X%(P1)<0 OR Y%(P1)<0 OR X%(P2)<0 OR Y%(P2)<0 THEN 790
777 IF X%(P1)>319 OR Y%(P1)>199 OR X%(P2)>319 OR Y%(P2)>199 THEN 790
780 SYS SL,X%(P1),Y%(P1),X%(P2),Y%(P2)
785 REM OSSZEKOTES
790 NEXT ZA:REM KOVETKEZO VONAL
900 POKE 198,0:WAIT 198,255:GET A$
910 SYS OF:LIST:REM GRAFIKA KI
1000 REM
1010 REM *****************
1020 REM ** KOORDINATAK **
1030 REM *****************
1100 DATA 10:REM PONTOK SZAMA
1110 DATA  0, 0, 0,  6, 0, 0,  6,10, 0
1120 DATA  0,10, 0,  3,15, 0,  3,15,15
1130 DATA  6,10,15,  6, 0,15,  0, 0,15
1140 DATA  0,10,15
2000 REM
2010 REM ****************
2020 REM ** OSSZEKOTES **
2030 REM ****************
2100 DATA 17:REM VONALAK SZAMA
2110 DATA  1, 2,  2, 3,  3, 4,  4, 1
2120 DATA  4, 5,  5, 3,  5, 6,  6, 7
2130 DATA  7, 3,  7, 8,  8, 2,  8, 9
2140 DATA  9, 1,  9,10, 10, 4, 10, 6
2150 DATA 10, 7
```



Ha kipróbájuk a programot, biztosan nem fogunk csalódni. Élvezet dolgozni vele. Az elkészült rajzokat tárolhatjuk, vagy készíthetünk róluk hardcopy-t.

A megrajzolt tárgyakat el is forgathatjuk. Ehhez a térkoordinátákat át kell számolnunk az új, elforgatott koordináta-rendszernek megfelelően (xr', zr', yr'), majd ezeket kell átszámítanunk síkkoordinátákká. A számoláshoz a következő képleteket használjuk:

Forgatás az xr tengely körül:

xr' = xr,

yr' = yr*cos (u)+zr*sin (u),

zr' = –yr*sin (u)+zr*cos (u).

Forgatás az yr tengely körül:

xr' = xr*cos (t)+zr*sin (t),

yr' = yr,

zr' = –xr*sin (t)+zr*cos (t).

Forgatás a zr tengely körül:

xr' = xr*cos (s)+yr*sin (s),

yr' = –xr*sin (s)+yr*cos (s),

zr' = zr,

ahol u, s és t az elforgatás szögét jelentik.

Ha az ábrát egyszerre két tengely körül akarjuk elforgatni, akkor először kiszámítjuk az egyik tengelyre vonatkozó elforgatási koordinátákat, majd ezeket behelyettesítjük a másik tengelyre vonatkozó képletekbe.

Háromdimenziós képet sok helyen alkalmaznak. Számítógéppel különösen jól, mérethelyesen ábrázolhatók a tárgyak és a környezetükhöz viszonyított helyzetük. A méretek, viszonyok könnyen módosíthatók, ezáltal rendkívül megkönnyítik épületek, gépek, berendezések tervezését, szerkesztését. (CAD = Computer Aided Design.) Autókat, repülőgépeket szerkesztenek, műszaki rajzokat készítenek a képernyőn. A megfelelő változatok papírra rögzíthetők. Ezenkívül a háromdimenziós grafika kiválóan alkalmas bonyolult folyamatok szimulációjára is.

5.1.2 Központos leképezés

Közismert dolog, hogy a "párhuzamosok a végtelenben találkoznak", ami azt jelenti, hogy a térben, a valójában párhuzamos egyenesek tőlünk távolodva közelítenek egymáshoz. Az előző programunkban ezt nem vettük figyelembe, ezért az elkészült ábrák egy kissé szokatlan képet nyújtottak. A valóságos látványt megközelítő ábrázolás bonyolult matematikai problémát jelent, amivel nem akarunk senkit terhelni. Egyszerűen csak megadjuk az f1, f2, f3 torzítási és az a, b, c eltolási tényezőket. Ebben az esetben a tengelyek merőlegesek egymásra és a képsíkot az x–y tengelypár alkotja. (Előző esetünkben az y és a z tengely helyzete fordított volt!)

x = (f1*xr+a)/q,

y = (f2*yr+b)/q,

ahol q = 1–(f3*zr+C)/fpz

xr, yr, zr:	térkoordináták,
f1, f2, f3:	x, y, z irányú torzítási tényezők,
a, b, c:	x, y, z irányú eltolási tényezők,
fpz:	iránypont távolsága (z koordináta).

A síkkoordináták kiszámításához először a q értékét kell kiszámítanunk. Tovább bonyolódik a helyzet, ha az ábrát el is akarjuk forgatni. Az egyenletek így alakulnak:

x = (f1*(A*xr+D*yr+G*zr)+a)/q,

y = (f2*(B*xr+E*yr+H*zr)+b)/q,

ahol q = 1-(f3*(C*xr+F*yr+I*zr)+C)/fpz.

Az A, B, C ... I változók jelentése a következő:

A = cos (s)*cos (t),
B = sin (s)*cos (t),
C = –sin (t),
D = –sin (s)*cos (U)+cos (s)*sin (t)*sin (U),
E = cos (s)*cos (U)+sin (s)*sin (t)*sin (U),
F = cos (t)*sin (U),
G = sin (s)*sin (U)+cos (s)*sin (t)*cos (U),
H = –cos (s)*sin (U)+sin (s)*sin (t)*cos (U),
I = cos (t)*cos (U),

ahol

s = a z tengely körüli elforgatás szöge,
t = az y tengely körüli elforgatás szöge,
U = az y tengely körüli elforgatás szöge.

Az ilyen hatalmas feladatok természetesen csak gépi nyelven oldhatók meg elfogatható gyorsasággal.

Időmegtakarítás céljából érdemes volna a szinusz–koszinusz táblázatot is – mondjuk fokonként – tárolni, hogy ne kelljen az értékeket állandóan kiszámítani, hanem a program adott részében egyszerűen betölthessük.

5.1.2.3 A háromdimenziós függvények

A háromdimenziós technika egyik csábító alkalmazási lehetősége a térbeli függvények ábrázolása. Ez nem csak a matematikusok szórakozását szolgálja, hanem nagyon is gyakorlati jelentőségű. Segítségével bonyolult, több táblázat adataiból is legfeljebb csak sejthető összefüggések szemléletesen és informatívan ábrázolhatók.

A kétdimenziós függvényábrázolás előnyei közismertek. Szinte az élet minden területén alkalmazzák ezeket. Tételezzük most fel azt, hogy valamilyen tényező függvényében több éven keresztül vizsgálni akarjuk egy termék árának alakulását. Ehhez minden évben egy kétdimenziós grafikont kell

készítenünk, ami az évek során görbék áttekinthetetlen halmazát eredményezné. Ekkor hívhatjuk segítségül a háromdimenziós ábrázolást, amellyel jó és gyors áttekintést kapunk a dolgokról és a kialakult tendencia figyelembevételével optimális döntéseket hozhatunk a jövőre vonatkozóan.

Az eljárás technikai megvalósítását matematikai függvényekkel szemléltetjük. A rajzoláshoz szükséges értékeket számítással határozzuk meg, de táblázatból is átvehetjük őket. Ezzel csak bővül a felhasználási lehetőségek sora.

A háromdimenziós grafikához egy ún. értéktáblázatot használunk és példaként a

$$z = x^2 + y^2$$

függvényt választjuk.

Emlékezzünk vissza az 5.1.1 fejezetre, ahol a függvényértékek kiszámítására egy FOR...NEXT-ciklust szerveztünk. Ebben tetszőleges számú x értékhez számítottuk ki a megfelelő y értékeket, meghatározva ezzel a görbe tetszőleges számú pontját.

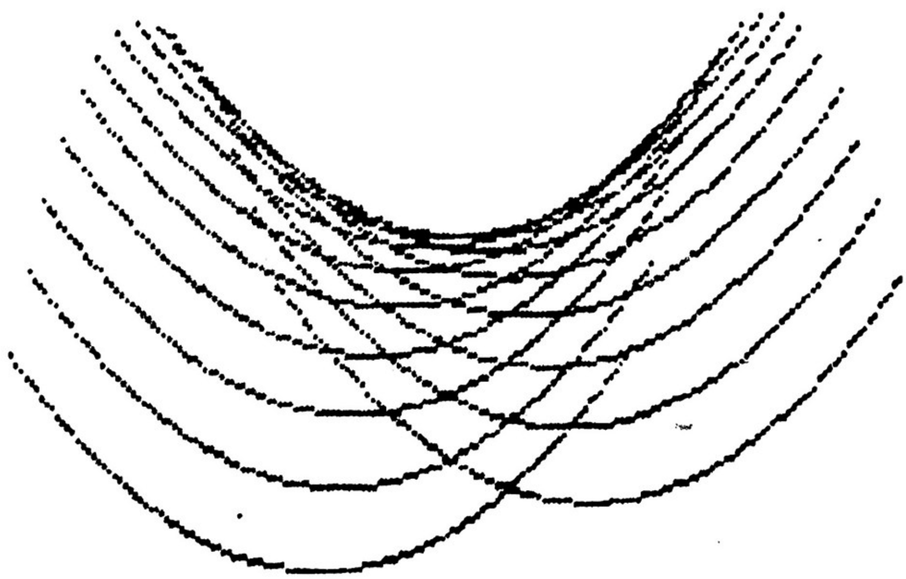
Háromdimenziós függvényeknél (f(x, y)) más a helyzet. Ebben az esetben két független változóból (x és y) kell kiszámítani a függvényértéket (z). Ehhez két, egymással szorosan összefüggő, FOR...NEXT-ciklust kell szerveznünk. Az elsőben az x értékeket, a másodikban az y-okat számláljuk. A következő program közelebb visz a probléma megértéséhez:

```
1 REM *** P 45 ***
2 REM
100 REM *********************
110 REM **                 **
120 REM ** 3-D: Z=Y↑2-X↑2 **
130 REM **                 **
140 REM *********************
150 REM
230 REM GEPI KODU ALPROGRAMOK
240 REM **********************
245 REM
250 IN=51200:SC=51203:REM ELOKESZITES/GRAFIKA KI
260 GC=51206:SC=51209:REM GR.TAROLO TORL./SZINEK
270 PC=51212:PL=51215:REM SZINEK MOD./PONT BE
280 UP=51218:SL=51221:REM PONT KI/VONAL RAJZOLAS
290 CL=51224:GL=51227:REM VONAL TORL./GR.LOAD
300 GS=51230:HC=51233:REM GR.SAVE/HARDCOPY
320 REM
330 REM PARAMETEREK
340 REM ***********
```

```
400 W = 3.1415/8
410 A = 0 : B = 0 : C = 0
420 F1=20 : F2= 5 : F3= 8
430 V1=160: V2=100
440 CO=COS(W):SI=SIN(W)
520 REM
530 REM RAJZOLAS
540 REM ********
600 SYS IN:SYS GC:SYS SC,16*1+4
610 FOR YR=3 TO -4 STEP -0.5
620 FOR XR=3 TO -3 STEP -0.05
630 ZR=YR↑2-XR↑2:REM A FUGGVENY
640 X=F1*(A+XR)+F3*(YR+C)*CO+V1
650 Y=F2*(B+ZR)+F3*(YR+C)*SI+V2
660 SYS PL,X,Y:REM PONTOK ELHELYEZESE
670 NEXT XR,YR
1000 POKE 198,0:WAIT 198,255:SYS OF:END
```

A szóban forgó FOR...NEXT-ciklusok a 610–670-es sorokban találhatók. A 630-as sorban történik a függvényérték, azaz a görbe pontjait kijelölő harmadik koordináta (z) kiszámítása. Ezután kerül sor a síkkoordináták kiszámítására (640–650-es sor).

Egyetlen problémánk – mint mindig – a paraméterek és az xr/yr változók értéktartományának kijelölése. Háromdimenziós függvényeknél a sok meghatározandó paraméter miatt ez fokozottan jelentkezik. Egyszerű számok behelyettesítésével segíthetünk magunkon. Tájékoztató értékek: a, b, c = 0; f1, f2, f3 = 1; w = PI/4; v1 = 160; v2 = 100. Az így kialakult képet ezután elképzelésünk szerint módosíthatjuk.



Térháló rajzolása:

Egy kis trükkel nagyon látványos dolgokat készíthetünk. Előző példánkban csak egy görbét ábrázoltunk, hogy magát a rajzolás folyamatát megfigyelhessük. Térhálós ábrákat egy ilyen görbe látszólagos elforgatásával érhetünk el. Azért mondjuk látszólagosnak, mert tulajdonképpen arról van szó, hogy a FOR...NEXT-ciklusban az xr és yr léptékeként különböző értékeket választunk, ami a képernyőn forgatásnak látszik. A fenti példában az xr ciklusban –0.5, az yr ciklusban pedig –0.05 a lépték. Így nem egyenletes felület jön létre, hanem valamilyen "csíkminta". Próbáljuk ki, mi történik, ha a két lépték egyenlő!

A léptékek felcserélésével a vonalak az előzőre merőlegesen rajzolódnak ki. Nézzünk erre is egy példát:

```
1 REM *** P 46 ***
2 REM
100 REM *********************
110 REM **                 **
120 REM ** 3-D: Z=Y↑2-X↑2 **
130 REM **                 **
140 REM *********************
150 REM
230 REM GEPI KODU ALPROGRAMOK
240 REM **********************
245 REM
250 IN=51200:SC=51203:REM ELOKESZITES/GRAFIKA KI
260 GC=51206:SC=51209:REM GR.TAROLO TORL./SZINEK
270 PC=51212:PL=51215:REM SZINEK MOD./PONT BE
280 UP=51218:SL=51221:REM PONT KI/VONAL RAJZOLAS
290 CL=51224:GL=51227:REM VONAL TORL./GR.LOAD
300 GS=51230:HC=51233:REM GR.SAVE/HARDCOPY
320 REM
330 REM PARAMETEREK
340 REM ***********
400 W = 3.1415/8
410 A = 0 : B = 0 : C = 0
420 F1=20 : F2= 5 : F3= 8
430 V1=160: V2=100
440 CO=COS(W):SI=SIN(W)
520 REM
530 REM RAJZOLAS
540 REM ********
600 SYS IN:SYS GC:SYS SC,16*1+4
610 SY=-0.5:SX=-0.03:REM LEPTEKEK
620 FOR ZA=1 TO 2:REM SZAMLALO
630 FOR YR=3 TO -4 STEP SY
640 FOR XR=3 TO -3 STEP SX
650 ZR=YR↑2-XR↑2:REM A FUGGVENY
```

```
660 X=F1*(A+XR)+F3*(YR+C)*CO+V1
670 Y=F2*(B+ZR)+F3*(YR+C)*SI+V2
680 SYS PL,X,Y:REM PONTOK ELHELYEZESE
690 NEXT XR,YR
700 IF ZA=1 THEN GOSUB 1100:SYS GC:SY=-0.04:SX=-0.5
710 NEXT ZA
720 FOR T=8192 TO 8192+8000
725 POKE T,PEEK(T) OR PEEK(T+8192):NEXT T
1000 POKE 198,0:WAIT 198,255:SYS OF:END
1100 FOR T=8192 TO 8192+8000
1105 POKE T+8192,PEEK(T):NEXT T:REM ATVITEL
1110 RETURN
```

Mint látjuk, a grafikát két különböző módon rajzoltuk meg. Miután az első elkészült, bekerült a \$2000–\$4000 (8192–16384) tártartományba. Ezután kerül sor a második rajzra a felcserélt léptékekkel. Amikor ez is készen van, a két grafikát OR-ral összekapcsoljuk és ezáltal a kettőből egy közös ábrát kapunk. Legyünk türelemmel, mert a tárolás és az összekapcsolás folyamata is rendkívül lassú!

A példában szereplő közbenső tárolás és utólagos OR-összekapcsolás nem feltétlenül szükséges. A második rajzot közvetlenül is rárajzolhattuk volna az elsőre. További példáinkban azonban már ez is elkerülhetetlen.

Ez a program is továbbfejleszthető például úgy, hogy a háromdimenziós koordináta-rendszert is felrajzoljuk a képernyőre.

Takart vonalak:

Ebben a részben ismertetünk egy viszonylag egyszerű eljárást, amellyel a függvények térbeli ábrázolásakor a "felület" látható részéről kitörölhetjük a nem látható rész vonalait.

A mindennapi életben általában nem látunk át a tárgyakon, eddig rajzaink viszont mind "átlátszóak" voltak. A felületet alkotó összes vonalat látjuk, holott az elülsőknek a hátsókat takarniuk kellene.

Függvények esetében két módszer létezik arra, hogy a nem látható vonalak megjelenését megakadályozzuk, ill. hogy azokat utólag kitöröljük.

Kezdjük az egyszerűbbel: azt kell megoldanunk, hogy a rajzolásnál hátulról előre, azaz csökkenő yr koordinátákkal haladjunk. Amikor egy pontot kiszámítottunk és a képernyőre rajzoltunk, az alatta levő többi pontot a képernyő aljáig kitöröljük. Így elérjük, hogy minden újonnan berajzolt sík takar

minden mögötte levőt, feltéve, hogy az a térben a berajzolt pont alatt van. Eredményként a térgörbe felülnézeti képét kapjuk. Legutóbbi programunkat a következő sorral kiegészítve minderről meg is győződhetünk:

```
685 SYS CL,X,Y+1,X, 199 : REM AZ ALSO PONTOK TORLESE
```

Ezzel az algoritmussal az alakzat alulnézetből nem rajzolható meg, mert amint azt a programfutás során látni fogjuk, az ehhez tartozó pontok mind törlődnek. Ezt úgy küszöbölhetjük ki, hogy nem a képernyő aljáig, hanem csak a következő vonalig törlünk. A rajzolás így persze sokkal tovább tart. Próbáljuk meg eszerint átírni a legutóbbi programunkat.

A következő néhány függvényt is érdemes kipróbálni, mert nagyon szép rajzokat készíthetünk velük:

$z = x^2 + y^2$,

$z = 1/(1 + x^2 + y^2)$,

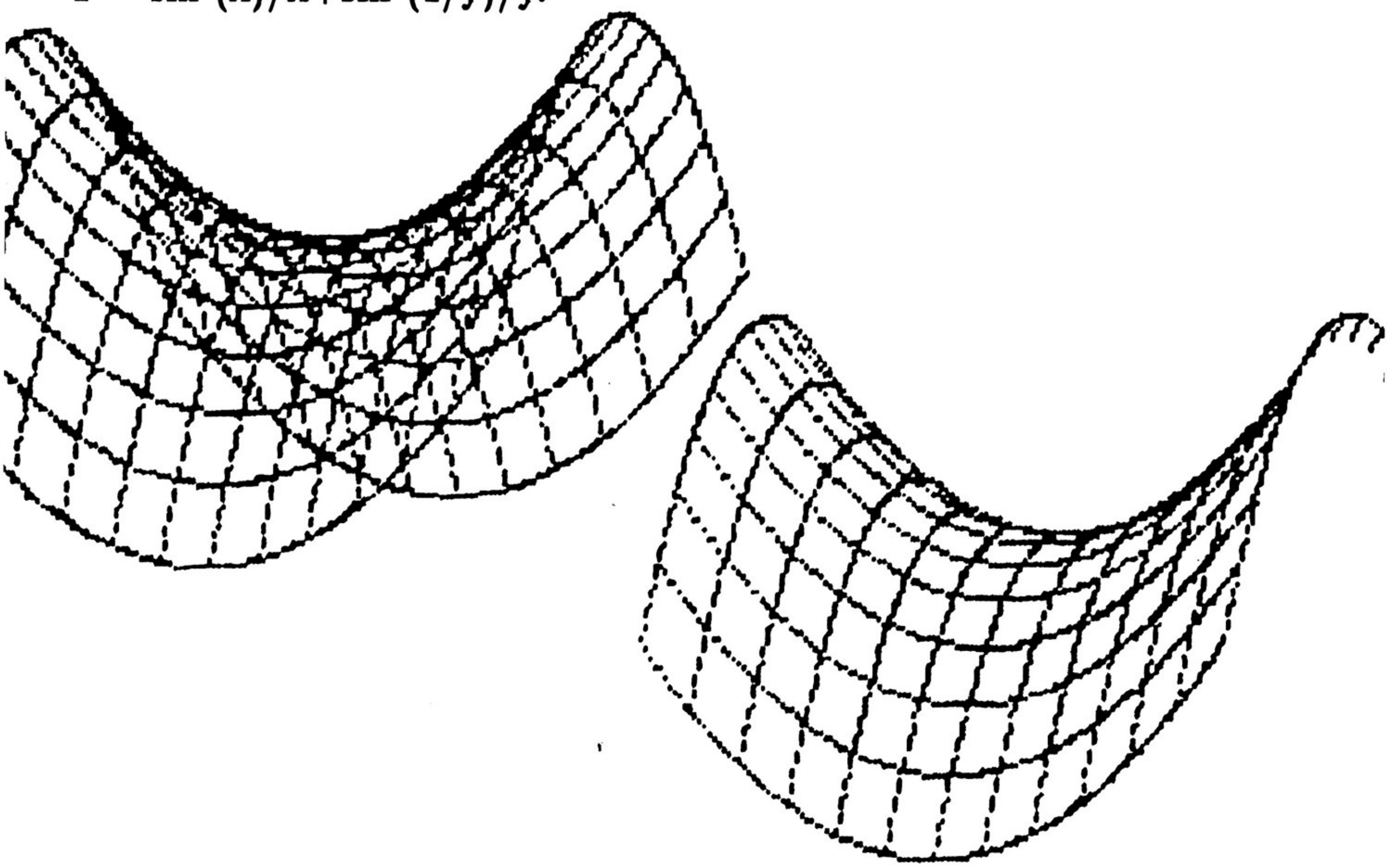
$z = SQR(1 - x^2/4 - y^2/9)$,

$z = \sin(x)/x + \sin(y)/y$,

$z = \sin(1/x)/x + \sin(1/y)/y$.

Ezek a függvények össze is keverhetők, például így:

$z = \sin(x)/x + \sin(1/y)/y$.



5.1.2.4 A képek mozgatása három dimenzióban

Legutóbbi fejezetünkben a háromdimenziós grafikák készítésének módszereivel ismerkedtünk meg. Tapasztaltuk, hogy a nagyobb ábrák rajzolása milyen sok időt vesz igénybe. Most foglalkozzunk egy kicsit azzal a kérdéssel, hogy elő lehet-e állítani a képernyőn mozgásnak tűnő folyamatokat? Elméletileg igen, és erre két módszer is létezik. Az első viszonylag egyszerű: két grafikai oldallal dolgozunk, amelyek felváltva jelennek meg. A két kép kismértékben eltér egymástól, ezért megfelelően gyors cserélgetés hatására mozgásként érzékelhető (ld. a 3.3.2 fejezetet is!). Gyors mozgások ábrázolására nem alkalmas, viszont az ábra forgatásával, ütemes nagyításával rendkívül látványos dolgokat idézhetünk elő.

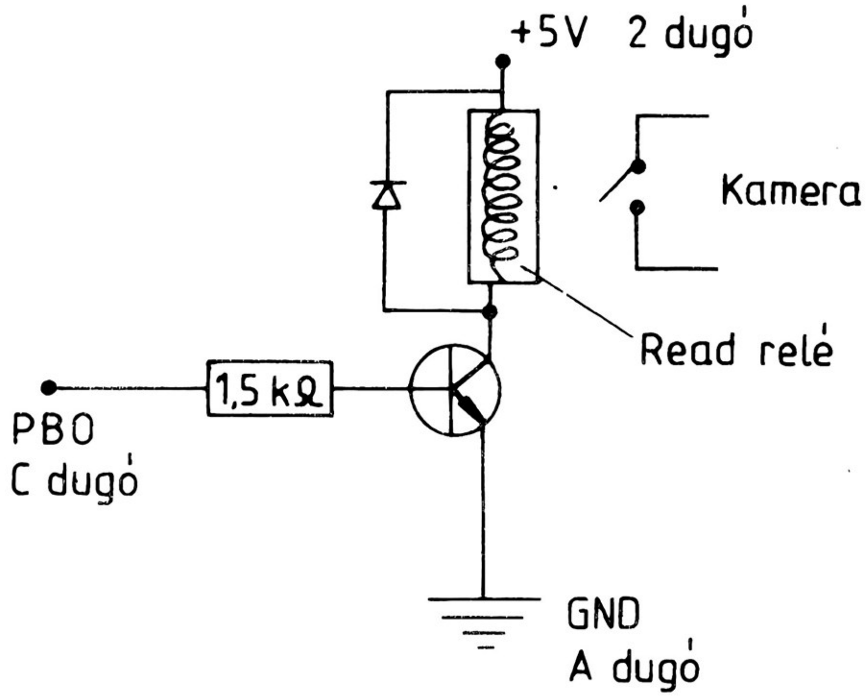
A másik módszer valamivel költségesebb, mert szükség van hozzá egy kamerára, amely egyedi képkapcsolóval és távkioldóval rendelkezik. A filmezéshez a rajz egy-egy fázisát a távkapcsoló pillanatnyi lenyomásával rögzítjük. Némi gyakorlattal nagyon szép eredményeket érthetünk el, és elkészíthetjük saját számítógép-grafikai filmünket, amilyet eddig csak moziban vagy tv-ben láthattunk.

A kamerát a tv-re kell rögzíteni (legjobb állvánnyal), majd beállítjuk az egyedi képkapcsolót és kioldjuk. Beprogramozzuk a képek sorrendjét. A kép elkészítésének ideje nem lényeges. Ha elkészült, lenyomjuk a távkioldót és rögzítjük. Ezt minél kisebb lépésekben célszerű végezni, hogy a filmen a folyamat ne legyen túl gyors. Kétes esetekben ismételjük meg a felvételt.

Ennél is elegánsabb megoldás, ha a kamerát a számítógép vezérli. Ez akkor lehetséges, ha a kamera elektromos távkioldóval is rendelkezik. Ha csak fémkioldónk van, a szaküzletekben szerezzünk be adaptert.

A következőkben ismertetünk egy eljárást, amellyel egyedi képkapcsoló nélkül is tudunk különlegesen rövid kapcsolási időt produkálni. Ehhez szükségünk van némi elektronikai ismeretre. A számítógép minden olyan esetben, amikor egy képet rögzíttetni akar a kamerával, egy jelet küld a felhasználói (vagy user-) portra.

A következő ábrán egy olyan kapcsolást közlünk, amivel a kamera a számítógéppel vezérelhető. Elkészítéséhez szükségünk van egy user-port-csatlakozóra és egy megfelelő csatlakozóra a kamera távkioldójához. A kereskedelemben mindkettő beszerezhető és a kapcsolást, esetleg egy szakember bevonásával, magunk is elkészíthetjük.



A kapcsolást kipróbáltuk és kiváló filmezési eredményeket értünk el. A kamera a következő utasítássorral vezérelhető:

```
10 C2 = 56576 : REM A CIA2 BAZISCIME ($DD00)
20 POKE C2+2, PEEK (C2+2) OR 1 : REM ERINTKEZO (PIN) =
   KIMENET
30 POKE C2,0 : REM KAMERA KI
40 POKE C2,1 : REM KAMERA BE
```

A 20-as sort a programfutás során csak egyszer kell megadni. Ügyeljünk arra, hogy a KI és BE művelet között elegendő idő legyen arra, hogy a relé és a kamera reagálhasson.

5.1.3 Grafikus statisztika

A grafika egyik közkedvelt alkalmazási területe a különböző, áttekinthetetlen táblázatok adatainak diagram formájában történő ábrázolás. Ezek közül legismertebbek a

- vonaldiagram,
- oszlopdiagram és a
- kördiagram.

a) Vonaldiagram

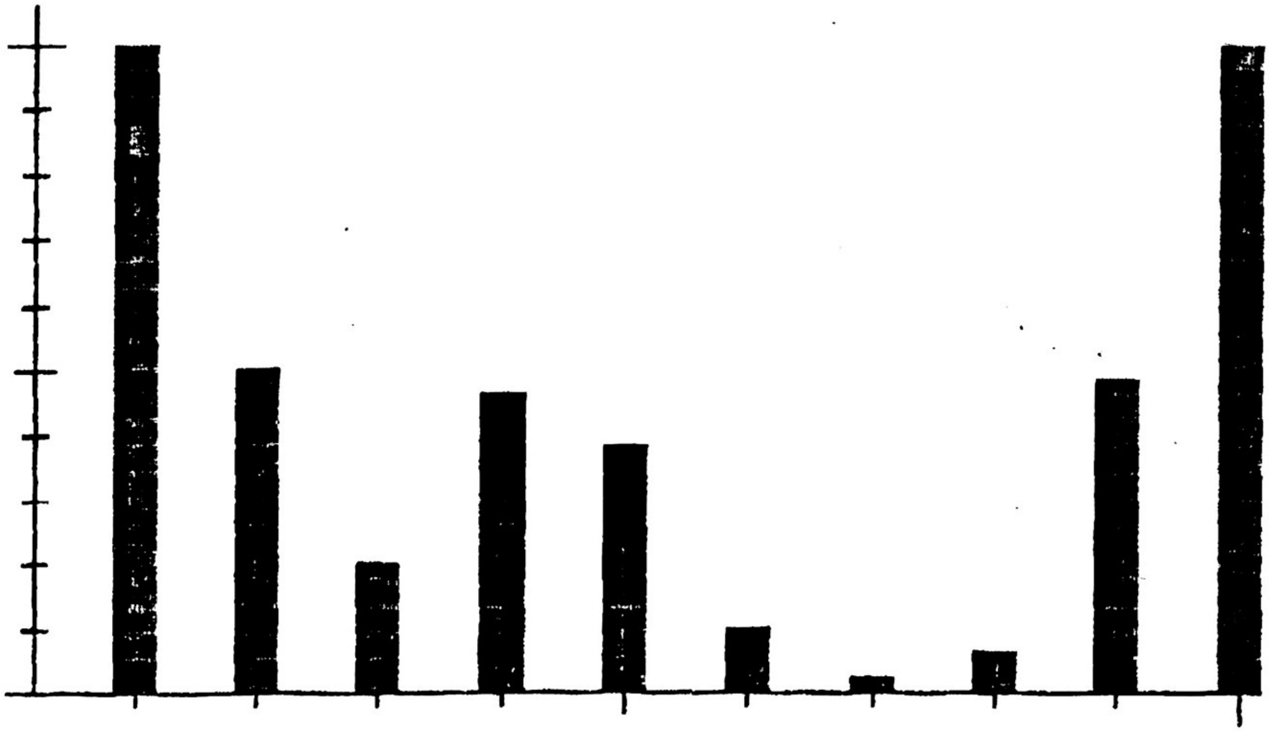
Ha egy meghatározott tényező függvényében több mérési eredményünk van (pl. a forgalom alakulása havi bontásban), az első diagramtípust használjuk. Az ábrázolás úgy történik, mint ahogy azt a kétdimenziós függvényeknél láttuk (5.1.1 fejezet). A különbség csupán annyi, hogy ebben az esetben nem számítással kapjuk a függvényértékeket, hanem táblázatból vesszük őket. Érdemes tehát még egyszer átnézni az említett fejezetet. A görbe eltolására, nagyítására és torzítására is az ott elmondottak érvényesek. Különösen lényeges azonban a koordinátatengelyek beosztása, ezért ezeket se hagyjuk el!

b) Oszlopdiagram

Az oszlopdiagramok ügye már nem ilyen egyszerű, bár az elv ugyanaz. Ezt az ábrázolási típust viszonylag kis adatállományok esetén alkalmazhatjuk. Az értékpár ebben az esetben nem a képernyő egy pontját jelöli ki, hanem egy oszlop x tengelyen elfoglalt helyét és magasságát. A következő program az oszlopdiagramok készítésére mutat példát. Tételezzük fel, hogy egy, a piaci viszonyoknak erősen kitett vállalat 10 hónapi forgalmát szeretnénk vizsgálni és az eredményt oszlopdiagramon kívánjuk szemléltetni. A mindenkori értékeket (1000 DM-ban) DATA-sorokban rögzítjük. Az adatok természetesen INPUT-tal is lekérdezhetők és akár mágneslemezre vagy kazettára is rögzíthetők. Máris van egy szép, kerek statisztikai programunk.

```
1 REM *** P 48 ***
2 REM
100 REM *********************
110 REM **                 **
120 REM ** OSZLOPDIAGRAM **
130 REM **                 **
140 REM *********************
150 REM
230 REM GEPI KODU ALPROGRAMOK
240 REM ***********************
245 REM
250 IN=51200:SC=51203:REM ELOKESZITES/GRAFIKA KI
260 GC=51206:SC=51209:REM GR.TAROLO TORL./SZINEK
270 PC=51212:PL=51215:REM SZINEK MOD./PONT BE
280 UP=51218:SL=51221:REM PONT KI/VONAL RAJZOLAS
290 CL=51224:GL=51227:REM VONAL TORL./GR.LOAD
300 GS=51230:HC=51233:REM GR.SAVE/HARDCOPY
320 REM
```

```
330 REM KOORDINATA TENGELYEK
340 REM ********************
350 SYS IN:SYS GC:SYS SC,16*7+10
360 READ ZA:REM AZ ERTEKPAROK SZAMA
370 READ HI:REM FELSO Y-ERTEKEK
380 XH=300:YH=180:REM A PONTOK MAX.SZAMA (X/Y IR.)
390 XE=INT(XH/ZA*1):YE=INT(YH/HI*10)
395 REM 1-ES ES 10-ES EGYSEGEK KISZMITASA
400 SYS SL,10, 10, 10,190:REM Y-TENGELY
410 SYS SL,10,190,310,190:REM X-TENGELY
420 T=-1:FOR Y=190 TO 10 STEP -YE
430 T=T+1:IF T/5-INT(T/5)=0 THEN SYS SL,5,Y,15,Y
432 REM KIEMELT BEOSZTAS
435 IF T/10-INT(T/10)=0 THEN SYS SL,3,Y,17,Y
440 SYS SL,7,Y,13,Y:REM EGYSEGEK BEJELOLESE
450 NEXT Y
460 T=0:BE=20+XE/2:REM ELSO VONAL KEZDOPONTJA
470 FOR X=BE TO 310 STEP XE
480 T=T+1:IF T/5-INT(T/5)=0 THEN SYS SL,X,195,X,190
485 IF T/10-INT(T/10)=0 THEN SYS SL,X,198,X,190
490 SYS SL,X,193,X,190
500 NEXT X
600 REM
610 REM DIAGRAM
620 REM *******
630 BR=XE-20:REM OSZLOPSZELESSEG KISZAMITASA
640 PO=BE-BR/2:REM ELSO OSZLOP KEZDO POZICIOJA
650 FOR T=1 TO ZA
660 READ DA:REM ADATOK BEOLVASASA
670 Y=190-DA*YE/10:REM OSZLOP MAGASSAGA
680 FOR X=PO TO PO+BR:REM SZELESSEGE
690 SYS SL,X,Y,X,190
700 NEXT X
710 PO=PO+XE:REM A KOVETKEZO OSZLOP HELYE
720 NEXT T
800 POKE 198,0:WAIT 198,255:SYS OF:END
900 REM
910 REM ADATOK
920 REM ******
1000 DATA 10:REM ERTEKEK SZAMA
1010 DATA 100:REM FELSO ERTEKEK
1100 DATA 100,50,20,46,38,10,2,6,43,99
```



Első látásra elég bonyolultnak tűnik, mert rengeteg képletet használtunk. Ezek azonban könnyen értelmezhetők, mert csupán a tengelyek és az oszlopok formálását végzik. Ebben a programban arra törekedtünk, hogy minél változatosabb grafikont készítsünk anélkül, hogy átlépnénk az értékhatárokat, vagy túl sok kis grafikont kellene készítenünk. Emiatt a tulajdonképpeni adatok elé még két érték került: az első az összes értékpár száma (10), a második a legnagyobb érték (100). Ezeket a 360/370-es sorok olvassák be. Ezután kijelöljük az oszlopdiagram által lefoglalt területet, ami most 300×180 pont (380-as sor).

Az első probléma a tengelyek megrajzolásakor és beosztásakor jelentkezik. Úgy állapodtunk meg, hogy az y tengelyen minden 10. egységet egy egységnyi, minden 50.-et egy dupla és minden 100.-at egy háromszoros hosszúságú vonallal jelölünk. A vízszintes tengelyre ugyanezt a beosztást visszük fel, azzal a különbséggel, hogy az 1., az 5. és a 10. egységeket emeljük ki. Ehhez először kiszámítjuk az egy (x tengely), ill. a tíz (y tengely) egységre eső pontok számát (390-es sor). Így sem jobbra, sem felfelé nem lépjük túl a tartományhatárokat, viszont az összes rendelkezésre álló felületet kihasználjuk.

Ezután következik a beosztás nélküli tengelyek megrajzolása (400/410-es sor), majd a beosztás (420/500-as sorok). Ez az eljárás viszonylag könnyen áttekinthető. A 430/435-ös és a 480/485-ös sorokban ellenőrizzük, hogy az 5. vagy 10. egységet jelöljük-e, és ha kell, megfelelően meghosszabbítjuk. Az IF mögötti kifejezés osztási maradékot ad, amit – funkcióját tekintve – az INT-utasítás ellentéteként foghatunk fel. A 460-as sorban levő képlet az oszlopok szélességével van összefüggésben és az első osztás helyét adja.

Most berajzoljuk az oszlopokat. Szélességüket úgy határozzuk meg, hogy elegendő hely maradjon közöttük (630-as sor). Az oszlopok kezdőpozícióját és magasságát meghatározó formulákat szintén meg kell érteni. (Az utolsó formula szerint a beolvasott adatot megszorozzuk az egységenkénti pontok számával, hogy az y tengely méretezésének megfeleljen.)

Az 1000-es sortól kezdődően tetszés szerint módosíthatjuk az adatokat, de arra ügyelni kell, hogy negatív és túl sok számot ne használjunk. Különlegesen nagy értékek esetén változtassuk meg a tengelyek méretezését, mértékegységét.

Ezzel némi áttekintést kaptunk az oszlopdiagramok programozástechnikájáról. Valószínűleg még számtalan változtatási lehetőség van, amelyekkel saját céljainkra fordíthatjuk a programot. Érdemes próbálkozni!

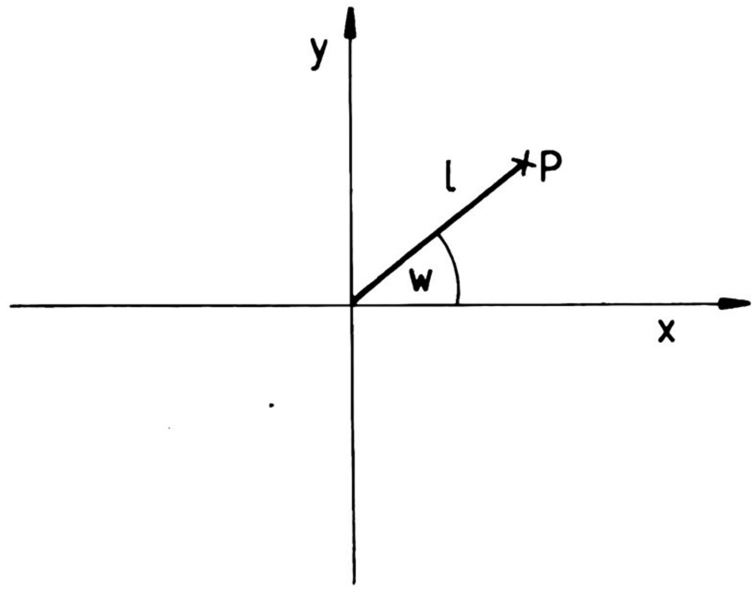
c) Kördiagram

Ez a módszer kifejezetten a mennyiségek el- és felosztásának, valamint a részmennyiségek arányának szemléletes ábrázolására alkalmas. Ehhez egy kört kell rajzolnunk, ami a felosztandó mennyiséget (100%) jelképezi, majd ezt különböző nagyságú cikkekre osztjuk az egészhez viszonyított százalékos arányban. Valószínűleg már mindenki találkozott ilyen diagrammal, így felhasználási területeiről nem kell sokat beszélnünk.

A kérdés inkább az, hogy hogyan kell egy ilyet programtechnikailag megvalósítani. Tudnunk kell, hogy a kör egy ilyen bonyolult relációjával állunk szemben. Az, hogy különböző részekre osszuk, nem egyszerű probléma és ráadásul a 4.2.2.3 fejezetben megismert általános köregyenletet sem tudjuk ebben az esetben alkalmazni. A metszetek ábrázolásához definiálnunk kell a szögeket is, ezért mint azt már említettük, néhány szóban ki kell térnünk a körábrázolás és az ún. polárkoordináták összefüggésére.

A függvényeket ún. polárkoordináta-rendszerben is ábrázolhatjuk. Ekkor egy pont helyzetét nem az x és y tengelytől mért távolságával jellemezzük, hanem egy, a síkban felvett ponttól (pólus) mért távolságával (1) és a pólust a ponttal összekötő egyenes és a vízszintes (polártengely) által bezárt szöggel.

A pólust és a polártengely irányát tetszőlegesen megválaszthatjuk, de a célszerűség azt kívánja, hogy a pólus egybeessen a derékszögű koordinátarendszer középpontjával, a polártengely pedig a x tengellyel. A következő ábrán ezt szemléltetjük:



A w szög változtatásával ezek szerint könnyen megrajzolhatjuk a kört. Az l ebben az esetben állandó. Általános ellipszisre a polárkoordinátákat (l, w) a következő összefüggések szerint számíthatjuk át derékszögű kordinátákká:

x = a*cos (w),

y = b*sin (w),

ahol

a – az ellipszis x irányú sugara,

b – az ellipszis y irányú sugara.

Ez a hozzárendelés már ismerős a 4.2.2.3 pontból. A képletek alapján tehát, csupán a szögkoordináta ismeretében, megadható az ellipszis egy területi pontja.

Még egy dologra ki kell térnünk. Már eddig is gyakran esett szó szögekről, de még nem tisztáztuk, hogy milyen formában kell megadnunk. Több lehetőségünk is van:

- bevitel régi fokban (0–360 fok),
- bevitel új fokban (0–400 fok),
- bevitel radiánban (0–2*Pi).

A hétköznapi életben az első a legelterjedtebb, számítógépünk viszont csak radiánban tud számolni (3. változat). A következő összefüggés áll fenn:

360 fok = 2*Pi (Pi = 3,1415.....).

Ez az érték megfelel egy egységnyi sugarú kör kerületének. A fok- és radiánértékek közti átszámításra a következő összefüggések érvényesek:

fok = 180*2*Pi/rad,

rad = 180*2*Pi/fok.

Ezek után nem nehéz megérteni a következő programot.

```
1 REM *** P 43 ***
2 REM
100 REM ****************
110 REM **            **
120 REM ** KORDIAGRAM **
130 REM **            **
140 REM ****************
150 REM
230 REM GEPI KODU ALPROGRAMOK
240 REM *********************
245 REM
250 IN=51200:SC=51203:REM ELOKESZITES/GRAFIKA KI
260 GC=51206:SC=51209:REM GR.TAROLO TORL./SZINEK
270 PC=51212:PL=51215:REM SZINEK MOD./PONT BE
280 UP=51218:SL=51221:REM PONT KI/VONAL RAJZOLAS
290 CL=51224:GL=51227:REM VONAL TORL./GR.LOAD
300 GS=51230:HC=51233:REM GR.SAVE/HARDCOPY
320 REM
330 REM ELLIPSZIS
340 REM *********
350 PI=3.1415
360 SYS IN:SYS GC:SYS SC,16*5+13
370 A=100:B= 60:V1=160:V2= 80
380 W=0:GOSUB 930:X1=X:Y1=Y:REM X1,X2ELOVALASZTAS
390 SP=7*PI/180:REM LEPTEK
400 BE=0:EN=2*PI:REM KEZDO- ES BEFEJEZO SZOG
410 GOSUB 800
420 BE=0:EN=1.03*PI:REM KB. 180 FOK
430 V2=100:REM MELYSEG
440 GOSUB 800:REM ELLIPSZIS RAJZOLASA
500 REM
510 REM KORCIKK
520 REM ******
530 READ ZA:DIM T(ZA):REM A RESZEK SZAMA
540 FOR S=1 TO ZA
550 READ T(S):REM ADATOK BEOLVASASA
560 SU=SU+T(S):REM OSSZEG KEPZESE
570 NEXT S
```

```
580 W=0:REM KEZDO SZOG
590 FOR S=1 TO ZA
600 PR=T(S)/SU:REM SZAZALEKKISZAMITASA
610 WA=2*PI*PR:REM RESZ SZOG
620 W=W+WA:REM VALODI SZOG
630 V2=80:GOSUB 930:REM KOORDINATAK KISZAMITASA
640 SYS SL,V1,V2,X,Y:REM OSZTOVONAL
650 IF W>PI THEN 690:REM CSAK A LATHATO OLDALON
660 X1=X:Y1=Y
670 V2=100:GOSUB 930:REM ALSO KOORD.-K KISZAMITASA
680 SYS SL,X1,Y1,X,Y
690 NEXT S
799 WAIT 198,255:SYS OF:END
800 REM
810 REM ELLIPSZISIV
820 REM ***********
830 FOR W=BE TO EN+SP STEP SP:REM SZOG MEGHATAROZ.
840 GOSUB 930:REM KOORDINATAK
850 SYS SL,X1,Y1,X,Y:REM VONAL
860 X1=X:Y1=Y
870 NEXT W:RETURN
900 REM
910 REM PONT ELHELYEZESE
920 REM ***************
930 X=A*COS(W)+V1
940 Y=B*SIN(W)+V2:RETURN
1000 REM
1010 REM ADATOK
1020 REM ******
1100 DATA 6:REM A RESZEK SZAMA
1110 DATA 20,10,15,40,30,8
```

A 800–940-es sorokban egy általános formulát találunk a kör vagy ellipszis pontjainak kiszámításához. A változók jelentése a következő:

- BE: az ellipszisív kezdőpontjának polárszöge,
- EN: az ellipszisív végpontjának polárszöge,
- SP: lépték, vagyis két egymás melletti pont szögtávolsága,
- V1: az ellipszis középpontjának x koordinátája,
- V2: az ellipszis középpontjának y koordinátája,
- A: az ellipszis x irányú sugara,
- B: az ellipszis y irányú sugara.

A léptékről (SP) annyit kell tudnunk, hogy a rajzon az ellipszis két egymás melletti kerületi pontjának szögtávolságát határozza meg, így befolyásol-

hatjuk vele a rajz pontosságát, finomságát. Két egymás melletti pontot mindig egyenessel kötünk össze (850-es sor). Megfelelően kis távolságok esetén látszólag egyenletes ívet kapunk. Speciális szögértékek választásával (pl. 30, 45... stb.) sokszöget rajzolhatunk, amely egyenlő oldalú lesz, ha a kiinduló síkidomunk kör.

Példaprogramunkat úgy írtuk meg, hogy az ábra háromdimenziós legyen. Nem egy körlapot, hanem egy vastagsággal rendelkező korongot rajzolunk, amelyből, mint a tortából, különböző vastagságú szeleteket vágunk. Az egészet felülről látjuk valamilyen szög alatt (nem merőlegesen!).

Először megrajzoljuk a korong felső lapját, egy ellipszist (410-es sor). Ezután megrajzoljuk ugyanennek az ellipszisnek az alsó felét, néhány ponttal lefelé eltolva. Ez a korong alsó lapja, aminek csak a fele látszik (440-es sor). Itt csalunk egy kicsit, mert a fél ívnél egy kicsit nagyobbat rajzolunk (w>1*Pi), így egyúttal a korong alkotói is elkészülnek. Igaz, ezek most nem tökéletes egyenesek, de a felbontás miatt ez nem feltűnő.

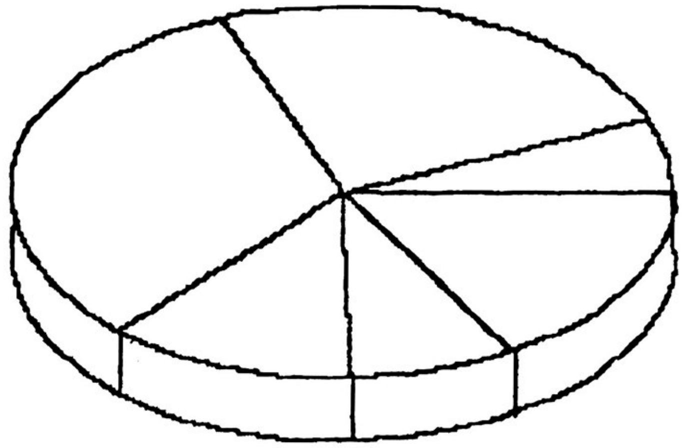
Most következik a "szeletelés":

Először néhány szót az adatokról, amelyeket DATA-sorban rögzítettünk (1110-es sor). Az első érték a cikkek számát jelenti, a többi a cikkek nagyságára vonatkozik. Ezek az értékek tetszőlegesek lehetnek.

Az 500-as sorban kezdődő rutin elején beolvassuk az adatok számát (ZA) és ennek megfelelően létrehozzuk a T(ZA) tömböt, amibe beolvassuk a cikkek nagyságára jellemző adatokat. A beolvasás ciklusban történik. A ciklusmagot az 550–560-as sorok képezik.

Magát a felosztást a következő FOR...NEXT-ciklus végzi. A 600-as sorban kiszámítjuk az adott rész egészhez viszonyított százalékos arányát, majd a 610-es sorban meghatározzuk az ennek megfelelő szöget, amit a 620-as sorban hozzáadunk az aktuális w polárszöghöz. Ekkor tehát már ismerjük a körcikknek megfelelő szöget, így kiszámíthatjuk az ellipszis hozzá tartozó kerületi pontjának helyzetét. Ezt a pontot összekötjük a középponttal (640-es sor). Most két eset lehetséges. Ha az ellipszis tőlünk távolabb eső felét osztottuk, akkor egyszerűen a következő "szelet" berajzolásával folytatjuk. Amikor viszont a felénk eső részt osztottuk, akkor az osztó vonalat függőlegesen folytatnunk kell az alsó ellipszisívig, hogy a térhatást érzékeltessük (650–680-as sor).

Ez a program is továbbfejleszthető. Kiszínezhetjük a cikkeket, vagy a legnagyobbat ki is emelhetjük stb.



5.2 Futó feliratok

Egy csábítóan egyszerű és mégis látványos dolog a tetszőleges alakú betűk mozgatása a képernyőn, azaz a betű-sprite-ok. Rendkívül jó a felbontásuk, ráadásul nagyíthatók és mozgathatók. Az okoz csak problémát, hogy BASIC-ben erre a célra nagyon kicsi tárterület áll rendelkezésünkre, mindössze négy blokk. Ez nagyon kevés ahhoz, hogy egy felirat összes betűjét tárolni tudjuk. Úgy segíthetünk magunkon, hogy a teljes VIC-címtartományt 16, 32 vagy 48 k-val feljebb toljuk (3.3.2 fejezet), és így olyan tártartományokba kerülünk, amelyeket minden további nélkül igénybe vehetünk. Tudnunk kell azonban, hogy ilyenkor nem csak a sprite-blokkok, hanem a video-RAM és a grafikustár is eltolódik.

Helyezzük át a sprite-blokkokat a $2000–$3FFF (8192–16383) címtartományba. Mivel ez egyben a grafikustár is, a grafikáról ebben az esetben le kell mondanunk. A sprite-ok definiálásához tehát összesen 127 (8192=128-tól 255-ig) blokk áll rendelkezésünkre. Ennek elegendőnek kell lennie, csak arra kell ügyelnünk, hogy ne írjunk túl hosszú programot, vagy olyat, amelyik a futás során túl nagy tárterületet használ.

Térjünk rá a betűkre. A sprite-mintákat DATA-sorokkal beépíthetnénk a programba is, de akkor a program túl nagy tárterületet igényelne. Ennél jobb módszerek is léteznek, amire a 4.3 fejezetben már kitértünk. Használhatjuk például a sprite-tervező segédprogramunkat (P23) és a mágneslemezre rögzített sprite-ot a program megfelelő helyén betölthetjük. Így egy komplett jelkészletet állíthatunk elő és használhatunk.

Példaprogramunkban mégis az előző megoldást választottuk, mert a szemléltetéshez néhány betű is elegendő lesz.

```
1 REM *** P 50 ***
2 REM
3 REM
4 REM
100 REM ******************
110 REM **              **
120 REM ** FUTO FELIRAT **
130 REM **              **
140 REM ******************
150 REM
160 V=53248:REM A VIC KEZDOCIME
170 POKE V+32,0:POKE V+33,0:REM KERET/HATTER FEKETE
175 PRINT CHR$(147)
180 REM
190 REM BETUK BETOLTESE
200 REM ***************
210 FOR X=1 TO 9: REM 9 BETU
220 READ A$:REM A BETU NEVE
230 BK=ASC(A$)-32+128:REM SPRITE-ASCII HOZZARENDELES
240 AD=BK*64:REM CIM 8192-TOL
250 FOR Y=AD TO AD+62
260 READ DA:POKE Y,DA
265 REM ADATOK BEOLVSASA ES BEIRASA
270 NEXT Y,X:REM 63 ADAT/8 BETU BEOLVASASA
300 REM
310 REM ELOKESZITES
320 REM ***********
330 FOR X=0 TO 7
340 POKE V+39+X,X+1:REM A 0-7 SPRITE SZINE
350 NEXT X
360 POKE V+23,255:REM MINDEN SPRITE NAGYITVA(Y)
370 POKE V+29,255:REM MINDEN SPRITE NAGYITVA(X)
380 POKE V+27,0:REM ELSOBBSEG
390 POKE V+28,0:REM NORMAL SZINEK
400 SP=0:REM ???
410 ZA=8:REM BETUK SZAMA A KEPERNYON (MAX.8)
420 AB=330/ZA:REM ???
430 GE=8  :REM SEBESSEG
500 REM
510 REM FUTO FELIRAT
520 REM ************
530 TE$="DATA-BECKER---":REM A FELIRAT SZOVEGE
540 FOR LA=1 TO 10:REM 10-SZERI ATFUTAS
550 FOR BU=1 TO LEN(TE$)
560 BU$=MID$(TE$,BU,1):REM FUTO BETUK
570 BK=ASC(BU$)-32+128:REM A BETU BLOKKSZAMA
580 POKE 2040+SP,BK:REM ???
590 POKE V+SP*2,95 :REM SPR. X-KOORD. ALSO BYTE
600 POKE V+SP*2+1,100:REM SPR. Y-KOORD.
610 POKE V+16,PEEK(V+16) OR 2↑SP
615 REM SPR. X-KOORDINATA, FELSO BYTE.
620 POKE V+21,PEEK(V+21) OR 2↑SP:REM SPR. BEKAPCS.
630 SP=SP+1:REM KOVETKEZO SPRITE SZAMA
635 IF SP=ZA THEN SP=0
640 FOR K=1 TO AB STEP GE:REM MOZGATAS
```

```
650 FOR S=0 TO ZA-1:REM ZA SPRITE MOZGATASA
660 AD=V+S*2:XL=PEEK(AD)-GE:XH=PEEK(V+16) AND 2↑S
665 REM A SPRITE X-KOORDINATAJA
670 IF XL<0 THEN XL=256+XL:POKE V+16,PEEK(V+16) AND 255-2↑S:IF
XH=0 THEN XL=0
680 POKE AD,XL
685 GOSUB 800
690 NEXT S,K
700 NEXT BU:REM KOVETKEZO BETU
710 NEXT LA:REM KOVETKEZO ATFUTAS
800 POKE V+21,PEEK(V+21) AND 255-2↑SP:RETURN
805 REM SPRITE KIKAPCSOLASA
1000 REM
1010 REM SPRITÉ-ADATOK
1020 REM ************
1090 REM
1100 DATA A: REM 'A' BETU
1110 DATA 000,000,000, 000,000,000, 003,255,192
1120 DATA 007,000,224, 006,000,096, 006,000,096
1130 DATA 006,000,096, 006,000,096, 006,000,096
1140 DATA 006,000,096, 007,255,224, 006,000,096
1150 DATA 006,000,096, 006,000,096, 006,000,096
1160 DATA 006,000,096, 006,000,096, 006,000,096
1170 DATA 006,000,096, 000,000,000, 000,000,000
1190 REM
1200 DATA B:REM 'B' BETU
1210 DATA 000,000,000, 000,000,000, 003,255,000
1220 DATA 003,001,128, 003,000,192, 003,000,192
1230 DATA 003,000,192, 003,000,192, 003,001,128
1240 DATA 003,255,000, 003,001,128, 003,000,192
1250 DATA 003,000,096, 003,000,096, 003,000,096
1260 DATA 003,000,096, 003,000,192, 003,001,128
1270 DATA 003,255,000, 000,000,000, 000,000,000
1290 REM
1300 DATA C:REM 'C' BETU
1310 DATA 000,000,000, 000,000,000, 001,255,192
1320 DATA 003,129,192, 003,000,000, 003,000,000
1330 DATA 003,000,000, 003,000,000, 003,000,000
1340 DATA 003,000,000, 003,000,000, 003,000,000
1350 DATA 003,000,000, 003,000,000, 003,000,000
1360 DATA 003,000,000, 003,000,000, 003,129,192
1370 DATA 001,255,192, 000,000,000, 000,000,000
1390 REM
1400 DATA D:REM 'D' BETU
1410 DATA 000,000,000, 000,000,000, 015,255,000
1420 DATA 012,001,128, 012,000,192, 012,000,192
1430 DATA 012,000,192, 012,000,192, 012,000,192
1440 DATA 012,000,192, 012,000,192, 012,000,192
1450 DATA 012,000,192, 012,000,192, 012,000,192
1460 DATA 012,000,192, 012,000,192, 012,000,128
1470 DATA 015,255,000, 000,000,000, 000,000,000
1500 DATA E:REM 'E' BETU
1510 DATA 000,000,000, 000,000,000, 003,255,192
1520 DATA 003,001,192, 003,000,000, 003,000,000
```

```
1530 DATA 003,000,000. 003,000,000. 003,006.000
1540 DATA 003,255,000. 003,006,000, 003,000.000
1550 DATA 003,000.000, 003,000,000. 003,000,000
1560 DATA 003,000,000, 003,000,000,003,001.192
1570 DATA 003,255,192, 000.000.000, 000.000.000
1590 REM
1600 DATA K::REM 'K' BETU
1610 DATA 000,000,000. 000,000,000. 003.001.128
1620 DATA 003,003,000, 003,006.000, 003,012.000
1630 DATA 003,024,000, 003,048.000. 003.096.000
1640 DATA 003,192.000, 003,192.000. 003.096,000
1650 DATA 003,048,000. 003,024.000. 003,012,000
1660 DATA 003,006,000, 003.003.000. 003,001,000
1670 DATA 003,000,192, 000.000,000. 000,000.000
1690 REM
1700 DATA R:REM 'R' BETU
1710 DATA 000,000.000, 000,000,000, 001,255.000
1720 DATA 003.003.128, 003,000.192, 003.000.192
1730 DATA 003,000.192. 003.000.192. 003,001,128
```

Mint minden eddigi programunk, ez is továbbfejleszthető a fantáziánktól függően. Gyors és mutatós csak gépi nyelven lesz, de így is jó szórakozást jelenthet. A sok REM-es sorral a program megértését szerettük volna megkönnyíteni.

Mint mondtuk, teljes jelkészletet szerkeszthetünk, amelyekből bármilyen futó felirat összeállítható.

Jó szórakozást kívánunk!

5.3 A játékok titka

A C 64-eshez több száz jó és kevésbé jó játékprogram kapható. Ha ezeket kipróbáljuk, meg lehetünk elégedve magunkkal, hiszen olyan számítógépet vásároltunk, ami mindent tud. Az összes program közül ugyanis a játékok azok, amelyek a készülék képességeit a legtökéletesebben bemutatják. Bizonyos esetekben már a megvalósíthatóság határát súrolják. De mit ér egy számítógép, ha a fantasztikus adottságait nem tudjuk kihasználni? Nem elég csak a mások által készített programokat alkalmazni, előbb-utóbb mindenki maga is készíteni akar valami hasonlót.

Ebben a könyvben sok mindenre fényt derítettünk és számtalan ötletet adtunk a grafikára, sprite-kezelésre és általában az optimális képernyőkijelzésre vonatkozóan. Gépünk azonban még ennél is több titkot rejt, ezért kísérletezzünk, kutassunk, próbálkozzunk bátran, mindig fedezhetünk fel új dolgokat.

A BASIC lassúsága miatt a legtöbb játékprogramot gépi nyelven írták. Mi is bemuttunk néhány rutint, amelyek assemblerben gyorsítják a programfutást és BASIC-utasításbővítésként foghatók fel (ld. grafikai programcsomag).

Ebben a fejezetben kifejezetten a játékok gyorsítását szolgáló technikai és bővítési lehetőségekre mutatunk példát, bár ezek más programokban is felhasználhatók. Segítségükkel BASIC-ben is viszonylag gyors és igényes játékokat programozhatunk.

5.3.1 Az animáció

Animáció alatt mozgó képek létrehozását értjük. A játékok többsége erre épül. Gyakran ez alapján sorolják be a játékokat az "akciójáték" és "egyéb" kategóriákba.

A C 64-as az animáció ötféle változatát különbözteti meg:

- sprite-ok helyzetváltoztatása,
- sprite-ok helyváltoztatása,
- jelek helyzetváltoztatása,
- jelek helyváltoztatása,
- grafikus animáció.

Helyzetváltoztatáson olyan mozgást értünk, amelynek során a sprite vagy jel képernyőn elfoglalt helye nem változik. Az első két eljárásra a 4.3.2 fejezetben már mutattunk példát. Olvassuk át még egyszer az ide vonatkozó részt, mert a játékoknál a sprite-programozásnak van talán a legnagyobb jelentősége.

Az utolsó módszerről nem akarunk sokat beszélni, mert a grafika elég lassú és ezért játékokban csak ritkán alkalmazzák.

Számunkra most a legérdekesebb a jelek hely- és helyzetváltoztatása. Ezt a módszert többnyire a jelkészlet módosításával együtt alkalmazzák.

a) A jelek helyzetváltoztatása

A módszer lényege ugyanaz, mint a sprite-ok esetében. Az egy vagy több jelből álló figura valamelyik részét több formában rögzítjük és ezeket folyamatosan váltogatva a képernyőn mozgást idézünk elő. Nézzünk egy példát: tételezzük

fel, hogy egy emberkét akarunk ábrázolni, amelyik mindkét kezét és lábát fel-le mozgatja. A figurát több részből állítjuk össze, hogy ne legyen túl kicsi. A mozgás létrehozásához több emberke mintáját kell rögzítenünk úgy, hogy egy-egy minta a mozgás egy-egy fázisát ábrázolja. Ezeket a képernyő azonos helyén, folyamatosan váltogatva jelenítjük meg, mozgást fogunk látni.

A jelek képernyőn elfoglalt helyének kijelöléséhez a 4.1 fejezetben bemutatott programszintű kurzorvezérlést használjuk. Mindez többszínű üzemmódban, egy módosított jelkészlettel vagy néhány különleges jellel igazán hatásos lehet. A játékok készítésénél a programozónak minden tudására és ügyességére szüksége van.

Példaprogramunkban az eredeti jelkészletet használtuk:

```
1 REM *** P 51 ***
2 REM
3 REM
4 REM
100 REM ******************
110 REM **              **
120 REM ** ANIMACIO - I **
130 REM **              **
140 REM ******************
150 REM
160 A$(0)="  "          +CHR$(119)
170 A$(1)=CHR$( 99)+CHR$(123)+CHR$( 99)
180 A$(2)=CHR$(167)+CHR$(183)+CHR$(165)
190 A$(3)="  "          +CHR$(113)
200 A$(4)=CHR$(173)+CHR$(123)+CHR$(189)
210 A$(5)=CHR$(183)+CHR$(183)+CHR$(183)
300 PRINT CHR$(147):REM KEPRNYO TORLES
310 X=18:REM OSZLOP HELYZETE
320 FOR ZA=0 TO 5 STEP 3
330 Y=12:GOSUB 1000:REM POZICIONALAS
340 PRINT A$(ZA):GOSUB 1010:REM FEJ
350 PRINT A$(ZA+1):GOSUB 1010:REM TORZS/KEZEK
360 PRINT A$(ZA+2):REM LABAK
370 FOR S=1 TO 100:NEXT S:REM VARAKOZAS
380 NEXT ZA
390 GOTO 320:REM KILEPES <RUN/STOP>-PAL
400 REM
410 REM POZICIONALAS
420 REM ************
1000 PRINT CHR$(19);:IF Y>0 THEN FOR T=1 TO Y:PRINT:NEXT T
1010 PRINT TAB(X);:RETURN
```

Először meghatározzuk, hogy az emberke mozgásának két fázisát melyik részek fogják ábrázolni (160–210-es sorok). Összesen hét jelből állítjuk össze, amelyek három egymás alatti sorban helyezkednek el. Mindazt, ami ebben a

három sorban van, egy tömbbe (A$(...)) rögzítjük. A 0...2 elemek az egyik, a 3...5 elemek pedig a második fázishoz tartoznak. Ezután meghatározzuk az emberke kezdőpozíciójának koordinátáit (330-as sor).

A további rész megértése már nem okoz problémát. Először megrajzoljuk az első fázist ábrázoló három sort, majd a FOR...NEXT-ciklus második lefutásában a második fázist. A 370-es sorban levő várakozási ciklus módosításával a mozgás sebességét változtathatjuk.

Próbáljuk meg az emberkét saját jelmintával ábrázolni (4.4 fejezet)! A mozgásnak több fázisát is rögzíthetjük, így még látványosabb lesz az előállított kép.

b) A jelek helyváltoztatása

Ugyanúgy, ahogy a sprite-okat, az előbbi emberkét is elmozdíthatjuk a képernyőn (ld. 4.3 fejezet). A képernyőn valahol megjeleníthetjük, majd egy pillanat múlva töröljük és a kiválasztott irányba egy kicsit eltolva újra megjelenítjük. A folyamatos eltolás a képernyőn mozgásnak látszik, ahogy azt a sprite-oknál már megbeszéltük. Egyetlen problémát csak a mozgás egyenletessége (felbontás) jelenthet. A figurát többnyire egy egész jellel el kell tolni. Ennél kisebb eltolás esetén közbülső fázisról beszélünk, amit csak jelkészlet-módosítással valósíthatunk meg.

Először nézzük az alapokat. A következő programban a két animációváltozatot együtt alkalmazzuk. Ha még azt is megoldjuk, hogy az emberke mozgása a botkormánnyal vezérelhető legyen, igazán büszkék lehetünk magunkra.

```
1 REM *** P 52 ***
2 REM
3 REM
4 REM
100 REM ******************
110 REM **              **
120 REM ** ANIMACIO - 2 **
130 REM **              **
140 REM ******************
150 REM
160 A$(0)="  "+"  "         +CHR$(119)+"  "       +"  "
170 A$(1)="  "+CHR$( 99)+CHR$(123)+CHR$( 99)+"  "
180 A$(2)="  "+CHR$(167)+CHR$(183)+CHR$(165)+"  "
190 A$(3)="  "+"  "         +CHR$(113)+"  "       +"  "
200 A$(4)="  "+CHR$(173)+CHR$(123)+CHR$(189)+"  "
210 A$(5)="  "+CHR$(183)+CHR$(183)+CHR$(183)+"  "
```

```
300 PRINT CHR$(147):REM KEPRNYO TORLES
305 SP=1:B=0:E=33
310 FOR X=B TO E STEP SP:REM OSZLOP
320 FOR ZA=0 TO 5 STEP 3
330 Y=12:GOSUB 1000:REM POZICIONALAS
340 PRINT A$(ZA)   :GOSUB 1010:REM FEJ
350 PRINT A$(ZA+1) :GOSUB 1010:REM TORZS/KEZEK
360 PRINT A$(ZA+2) :REM LABAK
370 FOR S=1 TO 60  :NEXT S:REM VARAKOZAS
380 X=X+SP
390 NEXT ZA
400 X=X-SP
410 NEXT X
420 SP=-SP:ZW=B:B=E:E=ZW:GOTO 310:REM???
500 REM POZICIONALAS
520 REM ************
1000 PRINT CHR$(19);:IF Y>0 THEN FOR T=1 TO Y:PRINT:NEXT T
1010 PRINT TAB(X);   :RETURN
```

Az előző programban használt mozgásrutint egy újabb FOR...NEXT-ciklusba ágyaztuk. A 160–210-es sorban a sor definícióját mindig egy szóközzel kezdjük és fejezzük be. Ezzel biztosítjuk az előző figura törlését mind előre-, mind pedig hátramozdulásnál. Próbáljuk megérteni az SP, B, E és ZW változók jelentését. Ezek tárolják a jobbra és balra mozduláshoz szükséges paramétereket.

Ezzel a néhány trükkel a sprite-ok mellett további eszközeink vannak a játékprogramozáshoz.

5.3.2 A gördítés

Mindenki látott már olyan játékot, amelyben a játékosnak valamilyen pályán (pl. autóversenypálya) kell eljutnia a célba, vigyázva arra, hogy közben a felbukkanó ellenfelek le ne szorítsák, le ne lőjék... stb. Ha ezeket a játékokat jobban szemügyre vesszük, kiderül, hogy nem is a játékos száduld vagy repül, hanem a háttér és annak elemei "jönnek" szembe vele. Ez a jelenség a képernyő, vagy a képernyő egy részének gördíthetőségén alapul. A gördítés jobbra, balra, fel- és lefelé történhet. Bonyolult és időigényes feladat, ezért csak gépi nyelven valósítható meg. Ilyenkor csak szöveg üzemmódban dolgozunk, mert grafikus üzemmódban egyszerre 8 k-t kellene mozgatnunk. Ez semmiféle hátrányt nem jelent, hiszen a jelkészlet módosításával szöveg üzemmódban is tudunk nagyfelbontású képet készíteni. Előnye viszont, hogy mindössze 1 k-t kelle mozgatnunk, ami lényegesen kevesebb munkával jár.

Példaként egy assembler-rutint mutatunk be, ami SYS-utasítással a program tetszőleges helyén lehívható.

```
P53

CC00                10          *= $CC00
CC00                20          ;
CC00                30          ;************************
CC00                40          ;**                    **
CC00                50          ;**     GORDITES       **
CC00                60          ;**                    **
CC00                70          ;************************
CC00                80          ;
CC00                90          ;
CC00               100          ;
B7F1               110 CHKGET   =$B7F1
07F6               120 FE       =2038
07F7               130 LE       =2039
00FD               140 FLAG     =$FD
00FE               150 SZAM     =$FE
0061               160 CIM      =$61
CC00 20 F1 B7      200 START    JSR CHKGET;EGESZ+BYTE OLVASAS
CC03 8A            210          TXA;JOBB/BAL JELZO
CC04 4A            220          LSR A
CC05 08            230          PHP
CC06 20 F1 B7      280          JSR CHKGET
CC09 E0 19         290          CPX #25;FELSO SOR
CC0B 90 02         300          BCC S1
CC0D A2 18         310          LDX #24
CC0F 8E F6 07      320 S1       STX FE
CC12 20 F1 B7      330          JSR CHKGET;ALSO SOR
CC15 E0 19         340          CPX #25
CC17 90 02         350          BCC S2
CC19 A2 18         360          LDX #24
CC1B 8E F7 07      370 S2       STX LE
CC1E 8A            380          TXA
CC1F AE F6 07      390          LDX FE
CC22 AC F7 07      400          LDY LE
CC25 38            410          SEC
CC26 ED F6 07      420          SBC FE;FEL-LE
CC29 B0 08         430          BCS S3
CC2B 49 FF         440          EOR #$FF;FEL<LE=>TAUSCH
CC2D AE F7 07      450          LDX LE
CC30 AC F6 07      460          LDY FE
CC33 85 FE         470 S3       STA SZAM;SZAMLALO
CC35 28            480          PLP
CC36 08            490          PHP;JOBB/BAL JELZO
CC37 90 03         500          BCC S4
CC39 C8            510          INY;JOBB SOR TOVABB
CC3A 98            520          TYA;ES ALSO INDUL
CC3B AA            530          TAX
```

```
CC3C BD CB CC    540 S4      LDA CIMF,X;CIM FELSO BYTE
CC3F 85 62       550         STA CIM+1
CC41 BD E5 CC    560         LDA CIMA,X;ALSO BYTE
CC44 85 61       570         STA CIM
CC46 28          580         PLP
CC47 08          590         PHP;JOBB/BAL JELZO
CC48 90 08       600         BCC MOVE
CC4A E9 01       610         SBC #1;JOBB-1 -NEL
CC4C 85 61       620         STA CIM
CC4E B0 02       630         BCS MOVE
CC50 C6 62       640         DEC CIM+1
CC52             650         ;
CC52             660         ;ELTOLAS
CC52             670         ;*******
CC52             680         ;
CC52 A5 62       690 MOVE    LDA CIM+1
CC54 29 03       700         AND #3
CC56 09 04       710         ORA #4;KEZDOCIM=$0400
CC58 28          720         PLP
CC59 08          730         PHP;JELZO(FLAG)
CC5A             740         ;
CC5A 20 86 CC    750         JSR MOVE1;VIDEORAM ELTOLASA
CC5D 28          760         PLP
CC5E 08          770         PHP;JELZO
CC5F A5 61       780         LDA CIM
CC61 90 0A       790         BCC M1
CC63 69 27       800         ADC #39;C=1!/JOBB
CC65 85 61       810         STA CIM
CC67 90 0C       820         BCC M2
CC69 E6 62       830         INC CIM+1
CC6B B0 08       840         BCS M2
CC6D E9 27       850 M1      SBC #39;C=0!/BAL
CC6F 85 61       860         STA CIM
CC71 B0 02       870         BCS M2
CC73 C6 61       880         DEC CIM
CC75 A5 62       890 M2      LDA CIM+1
CC77 29 03       900         AND #3
CC79 09 D8       910         ORA #$D8;SZINRAM $D800
CC7B 28          912         PLP
CC7C 08          915         PHP
CC7D 20 86 CC    920         JSR MOVE1;SZINRAM ELTOLASA
CC80 C6 FE       930         DEC SZAM
CC82 10 CE       940         BPL MOVE
CC84 28          950         PLP
CC85 60          960         RTS
CC86             970         ;
CC86             980         ;FELOSZTAS
CC86             990         ;*********
CC86             1000        ;
CC86 85 62       1010 MOVE1  STA CIM+1
CC88 90 03       1020        BCC BAL
CC8A 4C AB CC    1030        JMP JOBB
CC8D             1040        ;
```

```
CC8D              1050          ;ELTOLAS BALRA
CC8D              1060          ;*****************
CC8D              1070          ;
CC8D A0 00        1080 BAL      LDY #0
CC8F B1 61        1090          LDA (CIM),Y
CC91 AA           1100          TAX;ELSO BYTE
CC92 A0 27        1140          LDY #39
CC94 B1 61        1150 L2       LDA (CIM),Y
CC96 48           1160          PHA;MERKER1
CC97 8A           1170          TXA;MERKER2 OLVASASA
CC98 91 61        1180          STA (CIM),Y
CC9A 68           1190          PLA;MERKER1 OLVASASA
CC9B AA           1200          TAX;MERKER2-BE
CC9C 88           1210          DEY
CC9D 10 F5        1220          BPL L2
CC9F 18           1230          CLC
CCA0 A5 61        1240          LDA CIM
CCA2 69 28        1250          ADC #40;KOVETKEZO SOR
CCA4 85 61        1260          STA CIM
CCA6 90 02        1270          BCC L3
CCA8 E6 62        1280          INC CIM+1
CCAA 60           1290 L3       RTS
CCAB              1300          ;
CCAB              1310          ;ELTOLAS JOBBRA
CCAB              1320          ;**************
CCAB              1330          ;
CCAB 38           1340 JOBB     SEC
CCAC A5 61        1350          LDA CIM
CCAE E9 28        1360          SBC #40
CCB0 85 61        1370          STA CIM
CCB2 B0 02        1380          BCS R1
CCB4 C6 62        1390          DEC CIM+1
CCB6 A0 28        1400 R1       LDY #40
CCB8 B1 61        1410          LDA (CIM),Y;JOBB BYTE OLVASASA
CCBA AA           1420          TAX
CCBB A0 01        1460          LDY #1
CCBD B1 61        1470 R3       LDA (CIM),Y
CCBF 48           1480          PHA;MERKER1
CCC0 8A           1490          TXA;MERKER2 OLVASASA
CCC1 91 61        1500          STA (CIM),Y
CCC3 68           1510          PLA;MERKER1
CCC4 AA           1520          TAX;MERKER2-BE
CCC5 C8           1530          INY
CCC6 C0 29        1540          CPY #41
CCC8 D0 F3        1550          BNE R3
CCCA 60           1560          RTS
CCCB 04 04 04     1570 CIMF     .BYTE 4,4,4,4,4,4,4,5,5,5,5,5,5
CCD8 06 06 06     1580          .BYTE 6,6,6,6,6,6,6,7,7,7,7,7,7
CCE5 00 28 50     1590 CIMA     .BYTE $00,$28,$50,$78,$A0,$C8,$F0
CCEC 18 40 68     1600          .BYTE $18,$40,$68,$90,$B8,$E0
CCF2 08 30 58     1610          .BYTE $08,$30,$58,$80,$A8,$D0,$F8
CCF9 20 48 70     1620          .BYTE $20,$48,$70,$98,$C0,$E8
CCFF             65535          .END
```

```
ZEILEN:152   SYMBOLE:23   FEHLER:0

BAL    =CC8D  CHKGET=B7F1  CIM    =0061  CIMA   =CCE5  CIMF   =CCCB  FE     =07F6
FLAG   =00FD  JOBB  =CCAB  L2     =CC94  L3     =CCAA  LE     =07F7  M1     =CC6D
M2     =CC75  MOVE  =CC52  MOVE1  =CC86  R1     =CCB6  R3     =CCBD  S1     =CC0F
S2     =CC1B  S3    =CC33  S4     =CC3C  START  =CC00  SZAM   =00FE
```

A program BASIC-betöltő listája:

```
1 REM *** P 53 B ***
2 REM
3 REM
4 REM
100 FOR I=52224 TO 52480
110 READ X:POKE I,X:S=S+X:NEXT I
120 DATA  32,241,183,138, 74,  8
125 DATA  32,241,183,224, 25,144
130 DATA   2,162, 24,142,246,  7
135 DATA  32,241,183,224, 25,144
140 DATA   2,162, 24,142,247,  7
145 DATA 138,174,246,  7,172,247
150 DATA   7, 56,237,246,  7,176
155 DATA   8, 73,255,174,247,  7
160 DATA 172,246,  7,133,254, 40
165 DATA   8,144,  3,200,152,170
170 DATA 189,203,204,133, 98,189
175 DATA 229,204,133, 97, 40,  8
180 DATA 144,  8,233,  1,133, 97
185 DATA 176,  2,198, 98,165, 98
190 DATA  41,  3,  9,  4, 40,  8
195 DATA  32,134,204, 40,  8,165
200 DATA  97,144, 10,105, 39,133
205 DATA  97,144, 12,230, 98,176
210 DATA   8,233, 39,133, 97,176
215 DATA   2,198, 98,165, 98, 41
220 DATA   3,  9,216, 40,  8, 32
225 DATA 134,204,198,254, 16,206
230 DATA  40, 96,133, 98,144,  3
235 DATA  76,171,204,160,  0,177
240 DATA  97,170,160, 39,177, 97
245 DATA  72,138,145, 97,104,170
250 DATA 136, 16,245, 24,165, 97
255 DATA 105, 40,133, 97,144,  2
260 DATA 230, 98, 96, 56,165, 97
265 DATA 233, 40,133, 97,176,  2
270 DATA 198, 98,160, 40,177, 97
275 DATA 170,160,  1,177, 97, 72
280 DATA 138,145, 97,104,170,200
285 DATA 192, 41,208,243, 96,  4
290 DATA   4,  4,  4,  4,  4,  4
```

```
295 DATA   5,  5,  5,  5,  5,  5
300 DATA   6,  6,  6,  6,  6,  6
305 DATA   6,  7,  7,  7,  7,  7
310 DATA   7,  0, 40, 80,120,160
315 DATA 200,240, 24, 64,104,144
320 DATA 184,224,  8, 48, 88,128
325 DATA 168,208,248, 32, 72,112
330 DATA 152,192,232,  0,  0
340 IF S<>27098 THEN PRIN"HIBA A DATA-BAN !":END
350 PRINT "RENDBEN !"
```

Ez a rutin összhangban van a grafikai programcsomag rutinjaival (4. fejezet), tehát azzal együtt betölthető és használható. Lehívása a következő módon történik:

SYS 512224, r, a, e

ahol:

- r – a gördítés iránya (0=balra, 1=jobbr a),
- a – a gördítés első sora,
- e – a gördítés utolsó sora.

Alkalmazására próbáljuk ki a következő programot:

```
1 REM *** P 54 ***
2 REM
3 REM
4 REM
100 REM **************
110 REM **          **
120 REM ** GORDITES **
130 REM **          **
140 REM **************
200 SR= 52224:REM GORDITESI ALPROGRAM CIME
210 PRINT CHR$(147):REM KEPERNYO TORLES
220 PRINT:PRINT"A GORDITES AZT JELENTI,"
230 PRINT"HOGY A KEPERNYOSOROKAT "
240 PRINT"TETSZES SZERINT ELTOLHATJUK."
250 PRINT:PRINT"EGYSZERRE AKAR TOBBET IS!"
260 PRINT"[illegible]VALAHOGY IGY!"
270 FOR X=1 TO 5000:NEXT X
280 FOR X=1 TO 80
290 FOR Y=1 TO 50:NEXT Y
300 SYS SR,1,9,9
310 NEXT X:
```

```
315 FOR I=1 TO 2000:NEXT I:REM VARAKOZAS
320 PRINT:PRINT"GYORSABBAN !"
330 FOR X=1 TO 4000:NEXT X
340 FOR X=1 TO 200:SYS SR,1,11, 8:NEXT X
345 FOR I=1 TO 2000:NEXT I
350 PRINT"█VAGY VISSZAFELE?"
360 FOR X=1 TO 4000:NEXT X
370 FOR X=1 TO 400:SYS SR,0,13,13:NEXT X
375 FOR I=1 TO 2000:NEXT I
380 PRINT"█GORDITHETJUK AZ EGESZ KEPERNYOT IS!";
390 FOR X=1 TO 4000:NEXT X
400 FOR X=1 TO 200:SYS SR,0,0,24:NEXT X
405 FOR I=1 TO 2000:NEXT I
410 PRINT"█      IGY KESZUL A FUTO FELIRAT:"
430 FOR X=1 TO 4000:NEXT X
440 PRINT:PRINT"  -----DATA BECKER -----"
450 FOR X=1 TO 400:SYS SR,0,19,19:FOR Y=X TO 150:NEXT Y,X
```

Könyvünk keretein belül csak a játékprogramozás alapjainak megismertetésére volt módunk. A többit már mindenkinek magának kell megtapasztalnia és kikísérleteznie. Jelszónk: találékonyság és kitartás! Akinek sikerül elkészítenie első saját játékprogramját, ossza meg velünk is örömét! Sok sikert kívánunk!

6. FEJEZET

Függelék

6.1 Az optimális program

Könyvünkben nagyon sok BASIC-programot mutattunk be, amelyek akármilyen szépek is, bizony meglehetősen lassúak. Ez könnyen kedvét szegheti bárkinek, ezért most néhány olyan eljárással fogunk megismerkedni, amelyekkel a programjaink elegánsabbá, ügyesebbé tehetők.

A futási sebességen a következő módszerekkel javíthatunk:

a) a BASIC-program optimális szervezése,

b) a hosszú BASIC-rutinok helyettesítése assemblerrel.

a:

- Minél kevesebb REM-es sort használjunk, a ciklusoknál pedig mindenképpen mellőzzük.
- A program minél kevesebb sorból álljon. Különösen érvényes ez azokra a részekre, amelyeket a program gyakran használ. Rövidítsük az utasításokat, hogy minél több elférjen egy sorban.
- Ahol a program vagy a helyes írásmód nem követeli meg, ne használjunk szóközt.
- Az időkritikus ciklusokban csökkentsük a számításokat. Ezeket végezzük el a ciklus megkezdése előtt és csak az eredményt adjuk át egy közbenső változó alkalmazásával.
- A ciklusokban kerüljük a konstansokkal végzett közvetlen számításokat.
- Az időigényes ciklusokban lehetőleg kerüljük az alprogramlehívásokat (GOSUB, GOTO).
- Ciklust lehetőleg csak FOR...NEXT-tel szervezzünk, az IF-utasításokat kerüljük.

- A gyakran előforduló változókat a program elején definiáljuk. Különösen érvényes ez azokra, amelyek ciklusban is előfordulnak.
- Az összetartozó programrészeket lehetőleg egymáshoz közel helyezzük el, hogy a GOSUB- és GOTO-utasítások ugrócímeit a gépnek ne kelljen sokáig keresnie.
- A DATA-sorokat egy helyre írjuk és egy sorba minél több adat kerüljön.

Sajnos, ezek az intézkedések rontják a program áttekinthetőségét, ezért csak akkor végezzük el őket, ha a program már megfelelően működik. A gyorsítás mindig az utolsó művelet legyen.

b:

A gépi nyelvű rutinok alkalmazása nehezebb probléma, de nem matematikai folyamatok gyorsítására feltételenül érdemes elsajátítani.

A gépi betöltő program DATA-sorokból áll. Adatait a BASIC-program végrehajtása előtt READ-utasítással beolvassuk és POKE-utasítással a megfelelő tártartományba írjuk. Az így beírt program később SYS-szek hívható le. Erre nagyszerű példát láttunk a sprite-szerkesztő vagy jelszerkesztő segédprogramokban (P23, P30), de előnyei leginkább a grafikus programoknál érzékelhetők (grafikai programcsomag).

Itt a BASIC-ben igen hosszadalmas műveleteket (a grafikus tár törlése, színek beállítása) assembler-rutinokként írtuk meg:

a grafikustár törlése:

```
1 REM *** P 55 ***
2 REM
3 REM
4 REM
100 FOR I=51200 TO 51221
110 READ X:POKE I,X:S=S+X:NEXT I
120 DATA 169, 32,133,254,160, 0
125 DATA 132,253,162, 32,152,145
130 DATA 253,200,208,251,230,254
135 DATA 202,208,246, 96
140 IF S<> 3772 THEN PRINT"HIBA A DATA-BAN!"
150 PRINT"RENDBEN !"
```

a színek beállítása:

```
1 REM *** P 56 ***
2 REM
3 REM
4 REM
100 FOR I=51222 TO 51261
110 READ X:POKE I,X:S=S+X:NEXT I
120 DATA  32,241,183,134,151,162
125 DATA   3,169,  4,133,254,160
130 DATA   0,132,253,132,  2,165
135 DATA 151,145,253,200,196,  2
140 DATA 208,249,230,254,202,240
145 DATA   3, 16,242, 96,162,232
150 DATA 134,  2,208,235
160 IF S<>5970 THEN PRINT "HIBA ADATA-BAN!":END
170 PRINT"RENDBEN!"
```

Mindkét rutin beépíthető saját programjainkba. Lehívásuk a következőképpen történik:

SYS 51200 : REM A GRAFIKUS TAROLO TORLESE

SYS 51222,16*PSZ+HSZ : REM SZINEK BEALLITASA

ahol:	PSZ	– a pontok színének kódja	(0–15),
	HSZ	– a háttér színének kódja	(0–15).

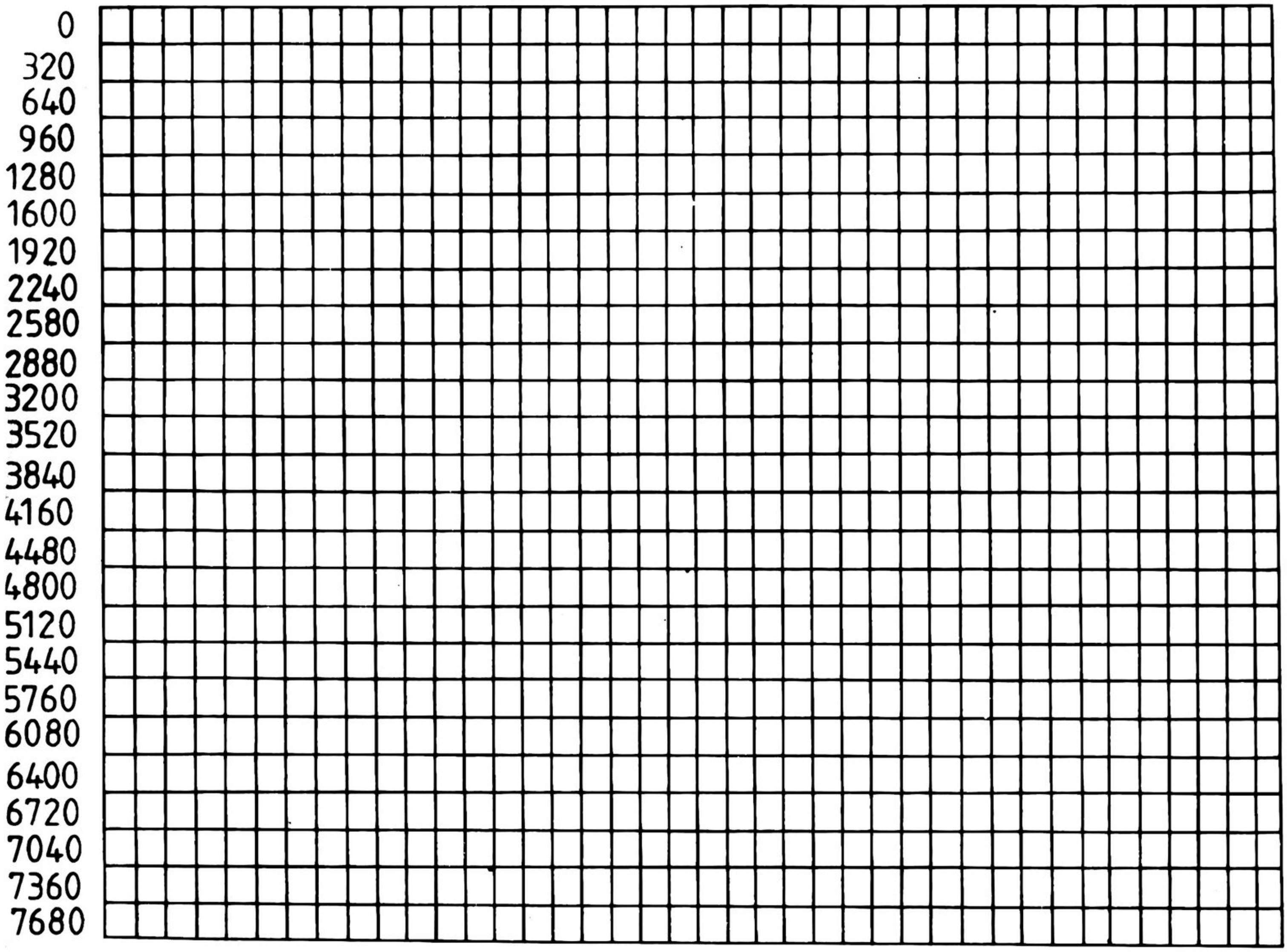
Erről a 4.7 fejezetben olvashatunk bővebben.

6.2 A grafikustár felépítése

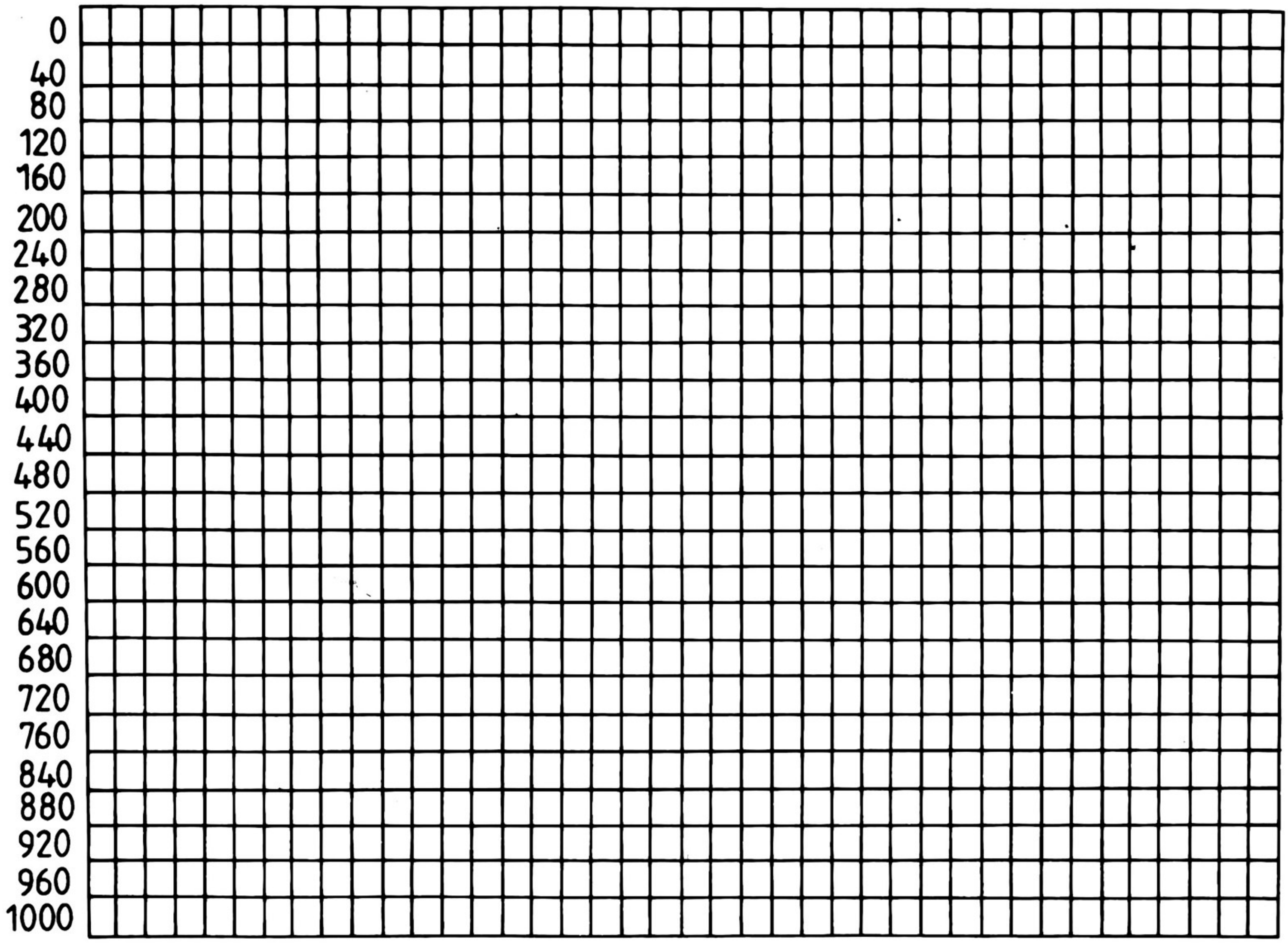
A következő ábra a grafikustár és a video-RAM felépítését szemlélteti. A sorok elején megadott relatív címekhez mindenkor hozzá kell adni a tár kezdő- vagy más néven báziscímét, ami a video-RAM-nál alapesetben $0400 (1024). A szín-RAM a video-RAM-hoz hasonlóan épül fel, kezdőcíme: $D800 (55296). A video-RAM kezdőcíme módosítható, azaz a video-RAM a táron belül eltolható.

A grafikustár minden helyét további 8×8 pontra kell bontanunk, hogy minden képernyői pontot szimbolizáljunk. Erre vonatkozóan olvassuk el még egyszer a 3. fejezetet.

6.2.1 A grafikustár



6.2.2 A video-RAM



6.3 Színkódtáblázat

A Commodore 64-es 16 szín előállítására képes. Minden színhez ún. színkódot rendeltek, amit POKE-utasítással írhatunk a megfelelő regiszterbe. Szöveg üzemmódban a jelek színét a billentyűzetről is kiválaszthatjuk. Az ennek megfelelő ASCII-kódokkal a szöveg színe a programban is előírható. A PRINT-utasításban ezek grafikus jelként jelennek meg.

Az összefüggéseket a következő táblázat szemlélteti:

Bill.	ASCII		Feladat	Kód		Szín
	Dec	Hex		Dec	Hex	
<ctrl> 1	144	$90		00	$00	fekete
<ctrl> 2	005	$05		01	$01	fehér
<ctrl> 3	028	$1C		02	$02	piros
<ctrl> 4	159	$9F		03	$03	türkiz
<ctrl> 5	156	$9C		04	$04	ibolya
<ctrl> 6	030	$1E		05	$05	zöld
<ctrl> 7	031	$1F		06	$06	kék
<ctrl> 8	158	$9E		07	$07	sárga
<C=> 1	129	$81		08	$08	narancs
<C=> 2	149	$95		09	$09	barna
<C=> 3	150	$96		10	$0A	világospiros
<C=> 4	151	$97		11	$0B	sötétszürke
<C=> 5	152	$98		12	$0C	középszürke
<C=> 6	153	$99		13	$0D	világoszöld
<C=> 7	154	$9A		14	$0E	világoskék
<C=> 8	155	$9B		15	$0F	világosszürke

6.4 Képernyőkódok

Minden képernyőn megjeleníthető jelnek ASCII- és képernyőkódja van. Ez utóbbi egy olyan szám, amellyel a jelet a video-RAM-ban rögzítjük. Amikor tehát egy jelet közvetlenül a képernyőtárba akarunk vinni, ezt az értéket egy POKE-utasítással a megfelelő rekeszbe írjuk. Az inverz jeleknek külön kódjuk van, amit úgy képezünk, hogy a normál jel kódjához 128-at adunk.

Kód	ASCII	Jele	
		1. sor	2. sor
0	64	@	@
1	65	A	a
2	66	B	b
3	67	C	c
4	68	D	d
5	69	E	e
6	70	F	f
7	71	G	g
8	72	H	h
9	73	I	i
10	74	J	j
11	75	K	k
12	76	L	l
13	77	M	m
14	78	N	n
15	79	O	o
16	80	P	p
17	81	Q	q
18	82	R	r
19	83	S	s
20	84	T	t
21	85	U	u
22	86	V	v
23	87	W	w
24	88	X	x
25	89	Y	y
26	90	Z	z
27	91	[	[
28	92	£	£
29	93	]	]
30	94	↑	↑
31	95	←	←
32	32		
33	33	!	!
34	34	"	"
35	35	#	#
36	36	$	$
37	37	%	%
38	38	&	&
39	39	'	'
40	40	(	(
41	41	)	)
42	42	*	*
43	43	+	+
44	44	,	,
45	45	-	-
46	46	.	.
47	47	/	/
48	48	0	0
49	49	1	1
50	50	2	2
51	51	3	3
52	52	4	4
53	53	5	5
54	54	6	6
55	55	7	7
56	56	8	8
57	57	9	9
58	58	:	:
59	59	;	;
60	60	<	<
61	61	=	=
62	62	>	>
63	63	?	?
64	96	─	─
65	97	♠	A
66	98	│	B
67	99	─	C
68	100	─	D
69	101	─	E
70	102	─	F
71	103	│	G
72	104	│	H
73	105	╮	I
74	106	╰	J
75	107	╯	K
76	108	L	L
77	109	╲	M
78	110	╱	N
79	111	┌	O

80	112		P
81	113		Q
82	114		R
83	115		S
84	116		T
85	117		U
86	118		V
87	119		W
88	120		X
89	121		Y
90	122		Z
91	123	+	+
92	124		
93	125		
94	126		
95	127		
96	160		
97	161		
98	162		
99	163		
100	164		
101	165		
102	166		
103	167		

104	168		
105	169		
106	170		
107	171		
108	172		
109	173		
110	174		
111	175		
112	176		
113	177		
114	178		
115	179		
116	180		
117	181		
118	182		
119	183		
120	184		
121	185		
122	186		
123	187		
124	188		
125	189		
126	190		
127	191		

6.5 Konverziótáblázat (dec/hex/bin)

		%	%	%	%	%	%	%	%	%	%	%	%	%	%	%	%
		0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
		0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
		0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
%0000 0000	0	$00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
%0001 0000	16	$10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
%0010 0000	32	$20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
%0011 0000	48	$30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
%0100 0000	64	$40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
%0101 0000	80	$50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
%0110 0000	96	$60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
%0111 0000	112	$70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
%1000 0000	128	$80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
%1001 0000	144	$90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
%1010 0000	160	$A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
%1011 0000	176	$B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
%1100 0000	192	$C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
%1101 0000	208	$D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
%1110 0000	224	$E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
%1111 0000	240	$F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

6.6 Sprite-tervező adatlap

Az itt bemutatott minta alapján elkészíthetjük magunknak a sprite-ok tervezéséhez szükséges adatlapot. A sprite minden pontját az adatlapon egy X-szel jelöljük és a bejelölt pontok értékét byte-onként összeadjuk. Ebből megkapjuk a sprite egy-egy sorát jellemző kódot (3–3 byte értéke).

Többszínű sprite-ok esetében két-két egymás mellett levő pont állapota (bekapcsolt/kikapcsolt) a pont színének származási helyére utal (ld. 3.5.2 fejezet).

bit:	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			
érték:	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	2	1	0	kódok		
0																											
1																											
2																											
3																											
4																											
5																											
6																											
7																											
8																											
9																											
10																											
11																											
12																											
13																											
14																											
15																											
16																											
17																											
18																											
19																											
20																											

6.7 Jeltervező adatlap

A sprite-tervező adatlaphoz hasonlóan használjuk.

bit :	7	6	5	4	3	2	1	0			
érték:	128	64	32	16	8	4	2	1	kódok		
7											
6											
5											
4											
3											
2											
1											
0											

6.8 A VIC regiszterei

regiszter		cím		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Dec	Hexa	Dec	Hexa	128	64	32	16	8	4	2	1
00	$00	53248	$D000	Sprite 0 — x-koordináta(Bits 0-7)(0-255)							
01	$01	53249	$D001	Sprite 0 — y-koordináta(0-255)							
02	$02	53250	$D002	Sprite 1 — x-koordináta (Bits 0-7)(0-255)							
03	$03	53251	$D003	Sprite 1 — y-koordináta(0-255)							
04	$04	53252	$D004	Sprite 2 — x-koordináta(Bits 0-7)(0-255)							
05	$05	53253	$D005	Sprite 2 — y-koordináta(0-255)							
06	$06	53254	$D006	Sprite 3 — x-koordináta(Bits 0-7)(0-255)							
07	$07	53255	$D007	Sprite 3 — y-koordináta(0-255)							
08	$08	53256	$D008	Sprite 4 — x-koordináta(Bits 0-7)(0-255)							
09	$09	53257	$D009	Sprite 4 — y-koordináta(0-255)							
10	$0A	53258	$D00A	Sprite 5 — x-koordináta(Bits 0-7)(0-255)							
11	$0B	53259	$D00B	Sprite 5 — y-koordináta(0-255)							
12	$0C	53260	$D00C	Sprite 6 — x-koordináta(Bits 0-7)(0-255)							
13	$0D	53261	$D00D	Sprite 6 — y-koordináta(0-255)							
14	$0E	53262	$D00E	Sprite 7 — x-koordináta(Bits 0-7)(0-255)							
15	$0F	53263	$D00F	Sprite 7 — Y-koordináta(0-255)							

Register		cím		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Dec	Hexa	Dec	Hexa	128	64	32	16	8	4	2	1
16	$10	53264	$D010	Sprite 0-7 — x -koordináta(Bit 8)(*256)							
				Sp. 7	Sp. 6	Sp. 5	Sp. 4	Sp. 3	Sp. 2	Sp. 1	Sp. 0
17	$11	53265	$D011	Raster Bit 8	bővített szín	Hi-Res-grafika	képernyő ki	25/24 sor	gördítés y irányban		
18	$12	53266	$D012	rasztersor (aktuális/választott) (Bit 0-7) (0-255)							
19	$13	53267	$D013	fényceruza — x-koordináta(0-255)							
20	$14	53268	$D014	fényceruza — y-koordináta(0-255)							
21	$15	53269	$D015		sprite be/ki						
				Sp. 7	Sp. 6	Sp. 5	Sp. 4	Sp. 3	Sp. 2	Sp. 1	Sp. 0
22	$16	53270	$D016	nem használt			több-színű	40/38 oszlop	gördítés x irányban		
23	$17	53271	$D017	sprite nagyítás y irányban							
				Sp. 7	Sp. 6	Sp. 5	Sp. 4	Sp. 3	Sp. 2	Sp. 1	Sp. 0
24	$18	53272	$D018	képernyőtár címe (*1024)				jelkészlettár címe (*2048)			nem használt
				Bit 13	Bit 12	Bit 11	Bit 10	Bit 13	Bit 12	Bit 11	
25	$19	53273	$D019	IRQ-Flag	nem használt			fény-ceruza	Sp.-Sp ütközés	Sp.-Hg. ütközés	raszter sorok
26	$1A	53274	$D01A	nem használt				fény-ceruza	Sp.-Sp ütközés	Sp.-Hg. ütközés	raszter sorok
27	$1B	53275	$D01B		sprite-háttérjel-elsőbbség						
				Sp. 7	Sp. 6	Sp. 5	Sp. 4	Sp. 3	Sp. 2	Sp. 1	Sp. 0
28	$1C	53276	$D01C		több színű sprite-ok						
				Sp. 7	Sp. 6	Sp. 5	Sp. 4	Sp. 3	Sp. 2	Sp. 1	Sp. 0
29	$1D	53277	$D01D		sprite nagyítás x irányban						
				Sp. 7	Sp. 6	Sp. 5	Sp. 4	Sp. 3	Sp. 2	Sp. 1	Sp. 0

regiszter		cím		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
dec	hex	Dez	Hex	128	64	32	16	8	4	2	1
30	$1E	53278	$D01E	Sprite-sprite-ütközés							
				Sp. 7	Sp. 6	Sp. 5	Sp. 4	Sp. 3	Sp. 2	Sp. 1	Sp. 0
31	$1F	53279	$D01F	Sprite - háttérjel - ütközés							
				Sp. 7	Sp. 6	Sp. 5	Sp. 4	Sp. 3	Sp. 2	Sp. 1	Sp. 0
32	$20	53280	$D020	keretszín (0-15)							
33	$21	53281	$D021	0. háttérszín (0-15)							
34	$22	53282	$D022	1. háttérszín (0-15)							
35	$23	53283	$D023	2. háttérszín (0-15)							
36	$24	53284	$D024	3. háttérszín (0-15)							
37	$25	53285	$D025	0. általános sprite-szín több színű üzemmódban (0-15)							
38	$26	53286	$D026	1. általános sprite-szín több színű üzemmódban (0-15)							
39	$27	53287	$D027	0. sprite színe (0-15)							
40	$28	53288	$D028	1. sprite színe (0-15)							
41	$29	53289	$D029	2. sprite színe (0-15)							
42	$2A	53290	$D02A	3. sprite színe (0-15)							
43	$2B	53291	$D02B	4. sprite színe (0-15)							
44	$2C	53292	$D02C	5. sprite színe (0-15)							
45	$2D	53293	$D02D	6. sprite színe (0-15)							
46	$2E	53294	$D02E	7. sprite színe (0-15)							